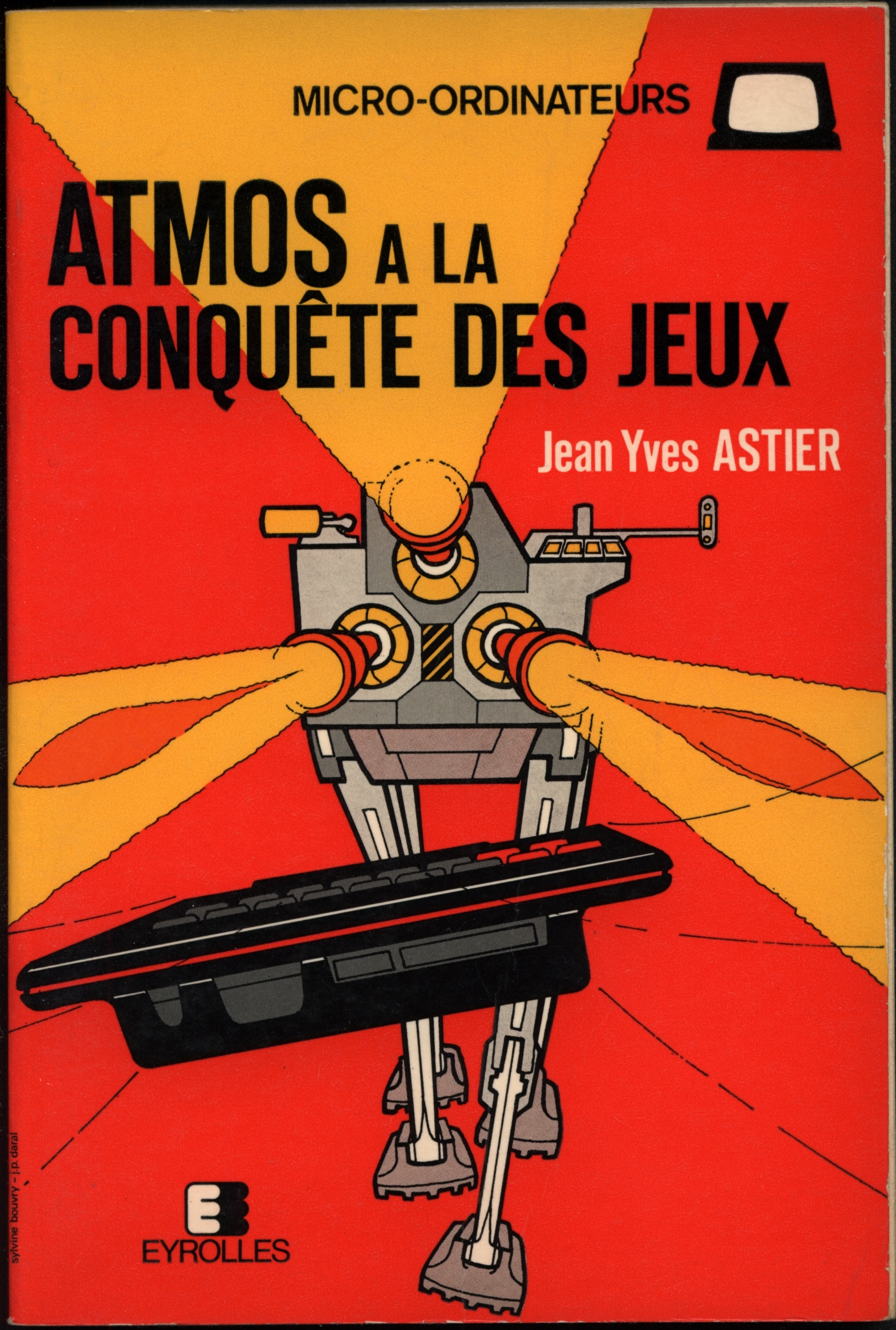

MICRO-ORDINATEURS
ATMOS A LA CONQUÊTE DES JEUX
Jean Yves ASTIER
EYROLLES
sylvine bouvry - j.p. daral

ATMOS
A LA CONQUÊTE
DES JEUX

ATMOS
A LA CONQUÊTE
DES JEUX

par

Jean-Yves ASTIER

Collection animée
par Richard SCHOMBERG



61, boulevard Saint-Germain — 75005 PARIS
1984

DANS LA MÊME COLLECTION

SCHOMBERG	- Le Basic Universel.
SCHOMBERG	- Micro-ordinateurs : Comment ça marche ?
HERNANDEZ	- Pascal par l'exemple.
NOLLET	- La conduite du ZX 81.
PELLIER	- La conduite du TRS 80.
LADEVIE	- Votre gestion avec BASIC sur micro-ordinateur.
QUEINNEC	- Langage d'un autre type : LISP.
PELLIER	- Programmez vos jeux d'action rapide sur TRS 80.
ASTIER	- La conduite de l'APPLE II.
	Tome 1 : le Basic de l'APPLE II.
	Tome 2 : le système graphique et l'assembleur de l'APPLE II.
MONTEIL	- L'assembleur facile du 6502 et du 6510.
LEPAPE	- L'assembleur facile du Z 80.
OROS et PERBOST	- ZX 81 à la conquête des jeux.
	- CASSETTE - ZX 81 à la conquête des jeux.
PERBOST	- CASSETTE N° 2 - 13 jeux 1 K.
DAX	- CP/M et sa famille.
NOLLET	- Langage machine, trucs et astuces sur ZX 81.
BICKING	- La conduite du PC 1212 (ou TRS 80 pocket).
TEJA	- Apprenez à parler à votre ordinateur.
MONTEIL	- La conduite du VIC 20.
PLOUIN	- La conduite de l'IBM-PC.
SAGUEZ	- Télécommande avec votre micro-ordinateur.
PELLIER	- Tout sur les disques du TRS 80 modèles I et III.
OROS et PERBOST	- La conduite du FX-702 P.
GROS	- La conduite du PC 1500.
HARWOOD	- Jeux et applications pour ZX SPECTRUM.
HARTNELL	- Le grand livre du ZX SPECTRUM.
HARTNELL et JONES	- La conduite du ZX SPECTRUM.
VULDY	- Graphisme 3 D sur votre micro-ordinateur.
BOUQUEROD	- Des extensions à construire pour votre ZX 81.
PINSON	- Le Basic en gestion sur Apple II.
WILLARD	- La conduite du TI 99.
CEYRAT	- Mon TI 99/4A.
DELANNOY	- Les fichiers en Basic sur micro-ordinateur.
AUBERT	- Pratiquez l'intelligence artificielle.
PERBOST et MASSE	- VIC 20 à la conquête des jeux.
TERRAL	- La conduite du T O7.
ASTIER	- La conduite de l'ORIC-1.
MONTEIL	- Premiers pas en LOGO.
MONTEIL	- La conduite du COMMODORE 64.
	Tome 1 : Basic, graphisme et son.
	Tome 2 : Langage-machine entrées/sorties et périphériques.
OROS	- La conduite de l'ATARI 400/800.
WILLARD	- TI 99 à la conquête des jeux.
KRUTCH	- Expériences d'intelligence artificielle en Basic.
ASTIER	- ORIC à la conquête des jeux.
PELLIER	- Langage machine, trucs et astuces sur ZX SPECTRUM.
SAGUEZ et ANDRIEUX	- Maîtrisez les interfaces de votre micro-ordinateur.
DE GEETER	- Forth par micros.
ASTIER	- ATMOS à la conquête des jeux.
CROWTHER et HARTLEY	- MO5 et TO7 à la conquête des jeux.
MONTEIL	- Introduction à l'IBM-PC Junior.

LOGILIVRES EYROLLES (logiciels sur cassettes)

Pellier	- Kamikaze (jeu pour ZX Spectrum).
Pellier	- Astéroïdes (jeu pour ZX Spectrum).
Pellier	- Othello/Isola (jeux pour ZX Spectrum).
Pellier	- Éditeur/Assembleur pour ZX Spectrum.
Perbost et Masse	- VIC 20 version de base à la conquête des jeux.
Haddadi	- Calcul des structures sur PC 1500/PC 2.

Si vous désirez être tenu au courant de nos publications, il vous suffit d'adresser votre carte de visite au :

Service « Presse », Éditions EYROLLES
61, Boulevard Saint-Germain,
75240 PARIS CEDEX 05,

en précisant les domaines qui vous intéressent. Vous recevrez régulièrement un avis de parution des nouveautés en vente chez votre libraire habituel.

Avant-propos

Grâce à ses incroyables capacités graphiques et sonores, votre ORIC est particulièrement doué pour conquérir le monde des jeux sur micro-ordinateurs. "ORIC/ATMOS à la conquête des jeux" se propose de vous guider dans cette jungle peuplée d'insectes voraces, de vaisseaux spatiaux, de couleurs et de sons étranges.

La première partie de cet ouvrage contient quinze jeux pour ORIC/ATMOS, jeux de réflexion ou d'action, qui bien sûr utilisent les dons musicaux et graphiques de votre micro-ordinateur préféré. Tous ces jeux fonctionnent sur n'importe quel ORIC/ATMOS standard, il n'est pas nécessaire de posséder un joy-stick.

La seconde partie contient des renseignements qui vous seront fort utiles pour programmer vos propres jeux, les trucs et les ficelles des modes graphiques, les adresses correspondantes. Vous saurez aussi comment est organisée la mémoire de l'ORIC/ATMOS, comment écrire un programme basic qui fabrique des lignes de programme basic, comment accélérer vos programmes et utiliser le langage machine.

Pour en savoir plus sur l'ORIC, je ne saurais vous recommander assez la lecture du livre "L'assembleur facile du 65Ø2", qui vous livrera tous les secret du micro-ordinateur 65Ø2, cerveau de votre micro-ordinateur.

Table des matières

1

Quelques jeux pour ORIC/ATMOS

1.1 GLOUTON

Le glouton est un petit insecte extrêmement vorace, que l'on trouve généralement dans les mémoires des micro-ordinateurs. Il se nourrit exclusivement de chiffres, qu'il avale goulûment. Pour ce jeu, le glouton se déplace sur votre écran, vous le dirigez grâce aux quatre touches flèches. Il faut amener l'insecte sur les chiffres qui apparaissent, en prenant bien soin d'éviter les étoiles, car le glouton ne les apprécie pas du tout.

Chaque chiffre dévoré vous donne autant de points, il faut bien sûr accumuler le plus de points possibles.

Au début du jeu, un nombre entre 1 et 2Ø vous est demandé. Il règle la vitesse d'apparition des chiffres et des étoiles. Frappez ce nombre, puis RETURN. A la fin, on vous demande si vous désirez rejouer. Frappez alors la touche "O" ou "N".

Avant que vous ne tapiez le programme, voici quelques explications sur son fonctionnement. Les lignes 1Ø à 12Ø sont des initialisations :

4Ø S est l'adresse-mémoire de la définition du caractère @, qui a 64 pour code ASCII. Pour faire apparaître le glouton, on va redéfinir @ pour qu'il figure l'insecte.

5Ø Ces huit nombres sont les valeurs des huit octets nécessaires pour redéfinir @ en glouton.

6Ø On place en mémoire les valeurs précédentes. Le caractère " @ " est remplacé par le caractère "glouton".

8Ø On fait passer l'écran en mode texte, s'il n'y était pas déjà, la couleur de l'écran est verte tandis que celle de l'encre est noire. Le nombre 64, qui est rangé dans la variable G, est le code ASCII du caractère "glouton". Ce code servira dans la suite du programme.

9Ø On lit un nombre compris entre 1 et 2Ø, qui est rangé dans F, puis on efface l'écran.

1ØØ Ceci élimine le "plop" qui retentit à chaque fois que l'on enfonce une touche. Le programme utilise les ordres PING et EXPLODE, aussi le "plop" est supprimé afin qu'il n'interrompe pas les bruits de clochette et d'explosion, lorsque vous frappez une touche.

Le jeu lui-même est constitué des lignes 13Ø à 32Ø. Ce bloc de programme est contenu dans une boucle qui débute à la ligne 14Ø et se termine en 32Ø. La boucle est exécutée 1ØØØ fois, après quoi le jeu est fini. Nous allons maintenant détailler ce qui se produit à l'intérieur de cette boucle.

15Ø 2Ø*RND(1) est un nombre aléatoire compris entre Ø et 19.9999999. Si ce nombre est plus petit que 2Ø-F, on se branche en 19Ø, sinon les lignes 16Ø à 18Ø seront exécutées. Si par exemple F vaut 1, 2Ø-F vaut 19 et le test est en moyenne vérifié 19 fois sur 2Ø. Si F vaut 19, 2Ø-F vaut 1 et le test sera vérifié 1 fois sur 2Ø en moyenne. En modifiant la valeur de F, on peut donc changer le nombre moyen de fois ou le test est vérifié.

Les lignes 16Ø à 18Ø font apparaître un chiffre ou une étoile en un point quelconque de l'écran.

16Ø Q est un nombre compris entre 2 et 39.9999999 tandis que R est lui compris entre Ø et 26.9999999. Le chiffre ou l'étoile apparaîtra à la position (Q, R) de l'écran, qui se trouve en mode texte.

17Ø C est compris entre 49 et 57.9999999. Puis, si RND(1)>.5, ce qui se produit en moyenne une fois sur deux, on place 17Ø dans C.

18Ø Il faut se rappeler que les fonctions SCRN et PLOT utilisent la **partie entière** de leurs arguments. Ainsi :

— Q donne un entier compris entre 2 et 39
— R donne un entier compris entre Ø et 26
— C donne un entier qui est:
  * soit compris entre 49 et 57 (code ASCII des chiffres 1 à 9)
  * soit égal à 17Ø (étoile en inverse vidéo).

G contient le code ASCII du glouton. Aussi, si le glouton ne se trouve pas en (Q, R), on y place le caractère qui a C pour code ASCII, c'est-à-dire un chiffre ou une étoile.

19Ø On se préoccupe de savoir si vous avez frappé une touche, on exécute la lecture du clavier A$=KEY$. Si aucune touche n'a été frappée, A$ est une chaîne vide et l'on va en 32Ø.

Si vous avez frappé une touche, normalement une flèche, on exécute les lignes 2ØØ à 31Ø qui déplacent le glouton suivant la touche.

2ØØ On place dans A le code ASCII de la touche frappée.

21Ø Si le code ASCII est celui de la touche "flèche à gauche" et si l'abscisse X du glouton est supérieure à 2, on décrémente cette dernière.

22Ø Si le code ASCII est celui de la touche "flèche à droite" et si l'abscisse X du glouton est inférieure à 39, on incrémente.

23Ø Si le code ASCII est celui de la touche "flèche en bas" et si l'ordonnée Y du glouton est inférieure à 26, on incrémente Y.

24Ø Si le code ASCII est celui de la touche "flèche en haut" et si l'ordonnée Y du glouton est supérieure à Ø, on décrémente Y.

26Ø On met dans V le code du caractère qui est en (X, Y) puis on place le glouton en (X, Y).

27Ø (X1, Y1) est la position précédente du glouton, où l'on place un blanc pour effacer le glouton précédent.

28Ø Si V est supérieur à 128, on a un caractère en inverse vidéo. On ôte 128 pour obtenir un code ASCII compris entre Ø et 127.

29Ø Si V vaut 42, on a rencontré une étoile et on se branche en 34Ø.

3ØØ Si l'on n'a pas rencontré un chiffre, de code ASCII compris entre 49 et 57, on va en 32Ø.

310 "1" a pour code ASCII 49, "2" est codé par 50, ... "9" est codé par 57. Ainsi V-48 vaut-il 1 si l'on a rencontré "1" sur l'écran, 2 si l'on a rencontré "2".

Après avoir exécuté le jeu, @ a été remplacé par le glouton. La ligne 410 réinitialise l'ORIC comme lorsque vous frappez "reset".

```
10 REM GLOUTON************************
20 :
30 REM DEFINITION BESTIOLE-----------
40 S=46080+8*64
50 DATA 33,18,12,12,12,12,18,33
60 FOR I=S TO S+7:READV:POKEI,V:NEXT
70 :
80 TEXT:CLS:PAPER 2:INK 0:G=64:P=0
90 PRINT:INPUT"Force (1-20) ";F:CLS
100 PRINT CHR$(6);
110 X=2:Y=2:X1=X:Y1=Y:PLOT X,Y,G
120 :
130 REM JEU-------------------------
140 FOR COUP=1 TO 1000
150 IF 20*RND(1)<20-F THEN 190
160 Q=2+38*RND(1):R=27*RND(1)
170 C=49+9*RND(1):IFRND(1)>.5THENC=170
180 IF SCRN(Q,R)<>G THEN PLOT Q,R,C
190 A$=KEY$:IF A$="" THEN 320
200 A=ASC(A$)
210 IF A=8ANDX>2 THEN X=X-1:GOTO 260
220 IF A=9ANDX<39 THEN X=X+1:GOTO 260
230 IFA=10ANDY<26 THEN Y=Y+1:GOTO 260
240 IFA=11ANDY>0 THEN Y=Y-1:GOTO 260
250 GOTO 320
260 V=SCRN(X,Y):PLOT X,Y,G
270 PLOT X1,Y1,32:X1=X:Y1=Y
280 IF V>128 THEN V=V-128
290 IF V=42 THEN 340
300 IF V<49 OR V>57 THEN 320
310 ZAP:P=P+V-48:GOTO 320
320 NEXT COUP:WAIT 300:GOTO 350
330 :
340 EXPLODE:WAIT 300
```

Note: le caractère "£" est à remplacer par "#".

```
350 CLS:PRINT:PRINT:PRINT:PRINT:PRINT
360 PRINT SPC(13);"Score:";P;CHR$(6)
370 PRINT:PRINT
380 PRINT"Voulez-vous rejouer (O/N) "
390 GET A$:IF A$="O" THEN 80
400 IF A$<>"N" THEN PING:GOTO 390
410 CALL DEEK(£FFFA)
```

Attention à ne pas oublier le ; de la ligne 1ØØ.

1.2 RÉFLEXE

Connaissez-vous bien la configuration du clavier de votre ORIC ? Ce petit jeu va permettre de le vérifier.

Au début du jeu, on vous demande le délai d'attente maximum. Tapez 2.5 puis RETURN. A chaque fois que ZAP retentit, un caractère apparaît quelque part sur l'écran. Il faut frapper la touche correspondante avant que le délai maximum ne soit écoulé. Vour remarquez que pendant le jeu le mot CAPS situé habituellement en haut à droite de l'écran a disparu. En effet, le clavier est en mode "minuscules", les touches alphabétiques donnent des lettres minuscules. Pour obtenir une lettre majuscule, il faut utiliser SHIFT, tout comme avec une machine à écrire. Lorsque vous serez familiarisé avec ce jeu, vous pourrez utiliser un délai plus court, par exemple 1.7.

Voyons maintenant comment fonctionne ce programme. Les lignes 1Ø à 17Ø servent à initialiser le jeu.

15Ø 32 est le code ASCII du caractère blanc. B$ est donc une chaîne constituée de trois caractères blancs, qui sera utilisée en 23Ø.

16Ø CHR$(17) supprime le curseur tandis que CHR$(2Ø) fait passer en minuscules.

Les lignes 18Ø à 24Ø ont pour but de tirer au hasard un caractère, et de l'afficher n'importe où sur l'écran.

19Ø On obtient l'abscisse X%, qui est un entier compris entre 3 et 36.

200 On obtient ensuite l'ordonnée Y%, entier compris entre 3 et 23.

210 C% est le code ASCII du caractère, compris entre 33 et 125.

220 La valeur 96 correspond au caractère ©, qui ne figure pas sur le clavier. On tire donc un nouveau code.

230 Cette boucle est exécutée trois fois, et remplit de blancs un carré de 3 × 3 caractères centré en (X%, Y%). En effet, la chaîne B$ contient 3 blancs.

240 Le caractère tiré au sort est affiché, en (X%, Y%). Comme l'on a effacé un carré autour de cette position à la ligne précédente, le caractère ne peut apparaître collé à un autre qui se trouvait déjà sur l'écran, ce qui simplifie le jeu.

De 260 à 280 on boucle en attendant que vous frappiez le caractère qui a été tiré. Si vous ne faites rien, on passe alors à la fin du programme, de 300 à 410, qui affiche le nombre de caractères trouvés.

320 Le caractère CHR$(10) permet d'obtenir des caractères double hauteur.

330 convertit le score S en une chaîne de caractères précédée de CHR$(32).

340 On élimine le caractère CHR$(32) situé en tête de la chaîne.

360 On calcule l'abscisse X où il faut afficher la chaîne pour qu'elle soit placée au milieu de l'écran.

370 On affiche deux fois la chaîne A$ pour obtenir des caractères double hauteur.

380 On rétablit le curseur et CAPS qui avaient été supprimés en 160.

```
10 REM REFLEXE************************
20 :
30 PAPER 2:INK 0
40 CLS:PRINT:PRINT:PRINT
50 PRINT"    Des caracteres ";
60 PRINT"apparaissent"
70 PRINT"au hasard sur l'ecran."
80 PRINT"    Il faut frapper le plus"
90 PRINT"vite possible sur la touche"
100 PRINT"correspondante."
```

```
110 PRINT
120 PRINT" -->les lettres majuscules"
130 PRINT"    s'obtiennent avec SHIFT"
140 PRINT:INPUT"Delai maximum ";DELAI
150 B=32:B$=CHR$(B)+CHR$(B)+CHR$(B)
160 S=0:CLS:PRINT CHR$(17);CHR$(20)
170 :
180 PING:WAIT 100
190 X%=2+INT(34*RND(1))
200 Y%=3+INT(21*RND(1))
210 C%=33+INT(93*RND(1))
220 IF C%=96 THEN 210
230 FORI=-1TO1:PLOTX%-1,Y%+I,B$:NEXT
240 PLOT X%,Y%,C%:ZAP
250 :
260 FOR I=1 TO DELAI STEP .002
270 IFKEY$=CHR$(C%)THENS=S+1:GOTO180
280 NEXT I
290 :
300 EXPLODE:WAIT 200
310 CLS:PAPER 1:INK 7
320 A$=CHR$(10)+"SCORE:"+CHR$(B)
330 SC$=STR$(S)
340 SC$=RIGHT$(SC$,LEN(SC$)-1)
350 A$=A$+SC$
360 X=18-INT(LEN(A$)/2)
370 PLOT X,11,A$:PLOT X,12,A$
380 PRINT CHR$(17);CHR$(20)
390 PRINT
400 PRINT"Il fallait frapper ";
410 PRINT CHR$(C%)
```

1.3 CAMÉLÉONS

Comme chacun le sait, les caméléons sont de petits animaux qui ont horreur de se déplacer. Pour se nourrir, ils se contentent d'attendre qu'un insecte passe devant eux, il n'y a plus alors qu'à déplier sa longue, longue langue pour happer l'insecte et en faire son repas.

L'ORIC vous demande tout d'abord : Force(1-20) ? Frappez par exemple 2 puis RETURN. Il apparaît alors 7 caméléons rangés à droite de votre écran. Il y a aussi un insecte. Le caméléon situé en face de l'insecte (qui est en fait un glouton) doit l'attraper. Vous devez frapper deux touches : le numéro du caméléon qui est situé face à l'insecte, puis la distance glouton-caméléon. Le numéro du caméléon est un numéro compris entre 1 et 7, 1 pour le caméléon rouge, 2 pour le vert, ..., 7 pour le blanc. La distance glouton-caméléon est comprise entre 1 et 9.

Ayant précisé qu'il ne faut pas frapper la touche RETURN après les deux chiffres, il ne me reste plus qu'à vous souhaiter un bon appétit !

Voici maintenant quelques explications sur la structure du programme. Tout d'abord, quelques initialisations, de 1Ø à 12Ø.

7Ø MAX est le délai d'attente maximum pour la frappe d'une touche. Cette valeur est utilisée en 19Ø et 22Ø.

8Ø On affiche les caméléons. Un caméléon est formé 4 × 2 caractères. Les quatre caractères supérieurs sont dans H$, que l'on affiche. On a affiché devant le caractère dont le code est I, qui définit la couleur.

9Ø On affiche comme précédemment le caractère de code ASCIII , qui définit la couleur. Puis les quatre caractères inférieurs du caméléon, qui sont dans P$, sont affichés.

Le jeu lui-même est une boucle exécutée quarante fois qui s'étend des lignes 14Ø à 43Ø.

En 15Ø-16Ø, on calcule la position (X, Y) de l'insecte, que l'on affiche à la ligne 17Ø.

De 19Ø à 21Ø, on attend que vous frappiez le numéro du caméléon, alors que de 22Ø à 24Ø, on attend la frappe de la distance insecte-caméléon.

De 26Ø à 28Ø, on vérifie que le numéro est bien compris entre 1 et 7, que la distance est bien comprise entre 1 et 9, puis ces deux quantités sont placées dans les variables A et B.

De 3ØØ à 39Ø, on déplie et on replie la langue du caméléon.

3ØØ Y1 est l'ordonnée de la langue, qui est horizontale.

31Ø La langue va se déplier de la droite vers la gauche, depuis X1=32 jusqu'à X1=32–3*B.

32Ø Produit une note.

33Ø L$ contient un caractère qui forme un élément de la langue.

34Ø La langue est dépliée. On note dans V le code ASCII du caractère situé juste à gauche de l'extrémité de la langue. Ce caractère est soit un blanc soit l'insecte.

35Ø La langue va se replier de X1=32–3*B à 32.

36Ø Identique à 32Ø.

37Ø On affiche V et un blanc, de code 32.

De 4ØØ à 42Ø, on regarde si le caractère attrapé est 115, c'est-à-dire l'insecte. Si oui, on incrémente le score. En 42Ø, on efface la position (X, Y), au cas où le caméléon n'aie pas capturé l'insecte, et on fait cesser tout bruit. Puis, on continue.

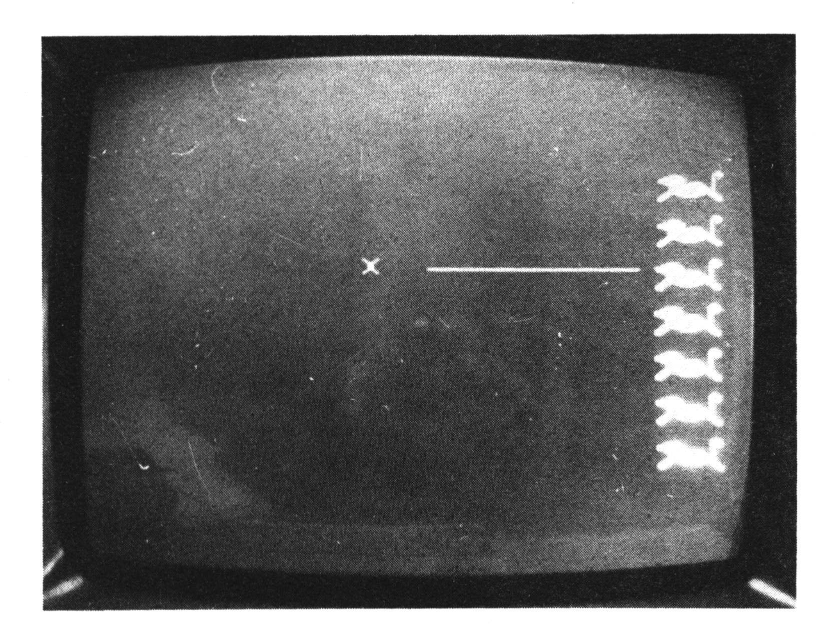
En 49Ø, l'instruction CALL #F8DØ recharge les deux jeux de caractères auxquels vous êtes habitué.

```
10 REM CAMELEONS**********************
20 :
30 TEXT:PAPER 0:INK 1:CLS:GOSUB 1000
40 :
50 REM TRACE CAMELEONS ET INITS------
60 CLS:INPUT"Force (1-20) ";F:CLS
70 MAX=3000/(F+5):FOR I=1 TO 7
80 PLOT 33,3*I,I:PLOT 34,3*I,H$
90 PLOT 33,3*I+1,I:PLOT 34,3*I+1,P$
100 NEXT I:SC=0
110 :
120 :
130 REM JEU--------------------------
140 FOR COUP=1 TO 40:Z=FRE("")
150 X=1+INT(9*RND(1)):X=3*X+1
160 Y=1+INT(7*RND(1)):Y=3*Y
170 PLOT X,Y,I$
180 :
190 FOR I=1 TO MAX
200 A$=KEY$:IF A$<>"" THEN 220
210 NEXT I:GOTO 420
```

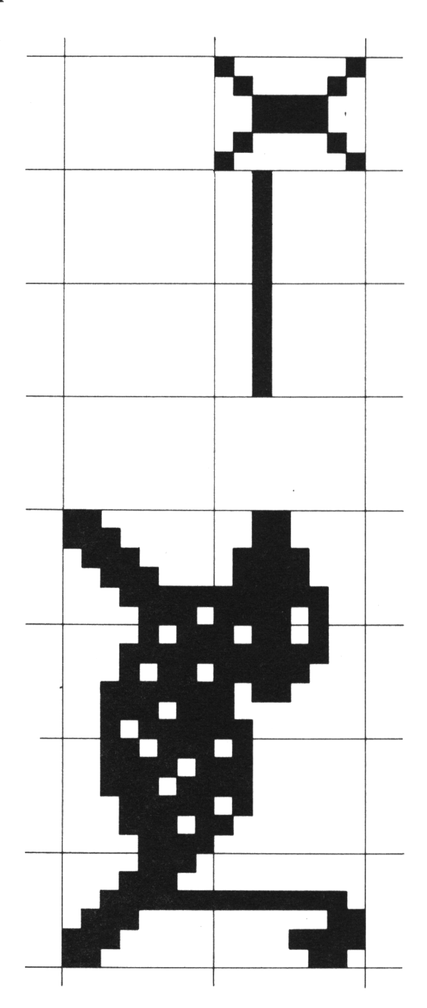
```
220 FOR J=1 TO MAX
230 B$=KEY$:IF B$<>"" THEN 260
240 NEXT J:GOTO 420
250 :
260 IF A$<"1" OR A$>"7" THEN 420
270 IF B$<"1" OR B$>"9" THEN 420
280 A=VAL(A$):B=VAL(B$)
290 :
300 Y1=3*A
310 FOR X1=32 TO 32-3*B STEP -1
320 SOUND 1,1000/(X1+3),0:PLAY1,0,7,0
330 PLOT X1,Y1,L$:NEXT X1
340 V=SCRN(X1,Y1)
350 FOR X1=32-3*B TO 32
360 SOUND 1,1000/(X1+3),0:PLAY1,0,7,0
370 PLOT X1,Y1,V:PLOT X1-1,Y1,32
380 NEXT X1
390 :
400 IF V<>115 THEN 420
410 PLOT 32,Y1,32:SC=SC+1:ZAP:WAIT 70
420 PLOT X,Y,32:PLAY 0,0,0,0
430 NEXT COUP
440 :
450 PLOT 12,12,"Score:"+STR$(SC)
460 PLOT 6,15,"On recommence (O/N) ?"
470 GET A$:IF A$="O" THEN 60
480 IF A$<>"N" THEN PING:GOTO 470
490 CLS:PAPER 2:INK 0:CALL £F8D0
500 END
510 :
1000 REM DEFINITION CARACTERES*******
1010 DATA 33,18,12,12,12,12,18,33
1020 DATA 0,0,0,0,0,63,0,0
1030 DATA 0,0,3,14,63,63,15,3
1040 DATA 0,0,48,24,60,60,25,63
1050 DATA 0,0,0,0,0,0,60,26
1060 DATA 6,15,11,10,8,8,8,8
1070 DATA 2,3,3,7,14,28,56,48
1080 DATA 55,63,29,55,30,7,0,0
1090 DATA 63,45,55,31,62,56,0,0
1100 DATA 8,40,56,56,28,14,7,3
1110 S=46080+115*8:F=S+10*8-1
1120 FOR I=S TO F:READ V:POKEI,V:NEXT
```

```
1130 I$=CHR$(115):L$=CHR$(116)
1140 H$=CHR$(117)+CHR$(118)+CHR$(119)
1150 H$=H$+CHR$(120)
1160 P$=CHR$(121)+CHR$(122)+CHR$(123)
1170 P$=P$+CHR$(124)
1180 RETURN
```

Note: le caractère "£" est à remplacer par "#".



Voici l'aspect du caméléon et de l'insecte :



1.4 CACHE-TAMPON

Je suis très fier de vous présenter CACHE-TAMPON, un jeu tout à fait original, comme vous allez pouvoir en juger :

— Pas de caméléon, de glouton, il n'y a rien à manger. Ce jeu est reposant pour l'estomac.

— Aucun dessin. On peut jouer sans regarder l'écran, vous pouvez même l'éteindre. Ce jeu est reposant pour les yeux.

— Aucune stratégie, tactique ou méthode, il n'y a pas à réfléchir. Ce jeu est reposant pour l'esprit.

Le jeu utilise le générateur programmable de sons de l'ORIC, qui est capable de produire simultanément trois notes. Imaginez que trois objets sont cachés derrière l'écran. A chacun de ces objets est associé une note, plus le curseur est proche de l'un des objets et plus la note associée est aiguë. Vous pouvez déplacer le curseur avec les touches flèches, il faut l'amener sur les objets cachés.

Tout d'abord, des lignes 1Ø à 9Ø, on trouve quelques initialisations.

3Ø CHR$(6) supprime le bruit produit par le clavier lorsque l'on enfonce une touche. En effet, ce bruit est aussi produit par le générateur programmable, il interromprait les trois notes.

4Ø 8 et 9 sont les codes ASCII des touches "flèche à gauche" et "flèche à droite". On les range dans les chaînes G$ et D$.

5Ø De même les codes ASCII des touches "flèche en bas" et "flèche en haut" sont placés dans les chaînes B$ et H$.

6Ø X et Y contiennent la position du curseur sur l'écran. X(3) et Y(3) servent à stocker la position des trois objets, tandis que F(3) est utilisé pour indiquer si les objets ont été trouvés ou non.

7Ø à 8Ø Les coordonnées des trois objets à découvrir sont tirées au hasard et stockées dans les deux tableaux.

La boucle de jeu s'étend des lignes 1ØØ à 28Ø. Il faut déplacer le curseur suivant la flèche qui a été enfoncée, regarder si le curseur ne se trouve pas sur l'un des objets, calculer la distance du curseur aux objets non trouvés, et produire les notes correspondantes.

11Ø On affiche le compteur, qui sera vert car on ajoute un caractère CHR$(2). CHR$(32) est un caractère blanc, qui sert à effacer le dernier chiffre affiché lorsque l'on passe de 1 ØØØ à 999, de 1ØØ à 99 et de 1Ø à 9.

12Ø On lit le clavier et l'on place le caractère lu dans K$.

13Ø Si l'on a frappé "flèche à gauche", on décrémente l'abscisse du curseur.

140 Si l'on a frappé "flèche à droite", on incrémente l'abscisse du curseur.

150 Si l'on a frappé "flèche en bas", on incrémente l'ordonnée du curseur.

160 Si l'on a frappé "flèche en haut", on décrémente l'ordonnée du curseur.

180 La touche lue est affichée, ce qui provoque le déplacement du curseur. Il ne faut surtout pas oublier le ; à la fin de l'ordre PRINT, qui empêche le renvoi du curseur au début de la ligne suivante.

190 Si F(I) vaut 1, cela signifie que l'objet numéro I a été découvert, on saute donc en 270, il n'y a rien à faire.

200 On calcule la distance du curseur à l'objet numéro I. La dernière instruction pourrait aussi s'écrire D%=SQR(A^2+B^2). Cette écriture est, de façon générale, à éviter. En effet, A*A et B*B sont calculés plus rapidement que A^2 et B^2. De plus, on obtient des résultats plus précis avec A*A et B*B, bien que dans le cas de ce programme cela soit sans importance.

210 Si la distance n'est pas nulle, l'objet n'est pas trouvé, on va en 250.

220 D% est nul, on vient de trouver l'objet I. On l'indique avec F(I)=1 et on incrémente le nombre N d'objets déjà trouvés.

230 L'expression logique F(1) AND F(2) AND F(3) est vraie si les nombres F(1), F(2) et F(3) sont tous les trois non nuls, c'est-à-dire si l'on a trouvé les trois objets. Dans ce cas, on va en 410.

250 SOUND définit la hauteur de la note I. La période de cette note est proportionnelle à D%, on obtiendra une note d'autant plus aiguë que D% est petit. Le paramètre "volume" de SOUND est à zéro : l'instruction définit la hauteur de la note sans la jouer.

260 M% est le minimum des distances curseur-objet. Ce nombre servira à déterminer la fréquence de répétition du son, en 270.

270 Pour jouer la note du canal 1, il faut exécuter PLAY 1, ... Pour la note du canal 2, on exécute PLAY 2, ... tandis que PLAY 4, ... active le canal 3. Si l'on désire obtenir simultanément les canaux 1 et 3, on emploie PLAY 1+4, ... tandis que PLAY 1+2, ... active les canaux 1 et 2 et PLAY 1+2+4, ... joue les 3 canaux.
A la ligne 180, on a exécuté P%=0. Puis on a, en 250, ajouté 1 s'il faut jouer le canal 1, on ajoute 2 s'il faut jouer le canal 2 et 4 s'il faut

jouer le canal 3. Ainsi, en 27Ø, P% contient la valeur adéquate pour que seuls les notes correspondants aux objets non trouvés soient jouées.

Les lignes 3ØØ à 4ØØ sont exécutées à la fin du jeu.

32Ø Il ne faut pas oublier le ;. Ainsi les messages des lignes 32Ø et 33Ø seront-ils sur la même ligne.

34Ø à 38Ø On reconnaît là l'indicatif de "Rencontres du 3e type".

39Ø CHR$(6) rétablit le bruit produit par le clavier, qui avait été supprimé à la ligne 3Ø.

Lorsque vous avez découvert les trois objets, on exécute les lignes 41Ø à 45Ø et l'on recommence le jeu.

```
10 REM CACHE-TAMPON******************
20 :
30 PAPER 1:INK 0:PRINT CHR$(6):CLS
40 W=1000:G$=CHR$(8):D$=CHR$(9)
50 B$=CHR$(10):H$=CHR$(11)
60 X=2:Y=0:DIM X(3),Y(3),F(3)
70 FOR I=1 TO 3:X(I)=INT(38*RND(1))+2
80 Y(I)=INT(27*RND(1)):NEXT
90 :
100 FOR T=W TO 0 STEP-1
110 PLOT16,13,CHR$(2)+STR$(T)+CHR$(32)
120 K$=KEY$:IFK$=""THENWAIT10:GOTO280
130 IF K$=G$ANDX>2 THEN X=X-1:GOTO180
140 IF K$=D$ANDX<39 THENX=X+1:GOTO180
150 IF K$=B$ANDY<26 THENY=Y+1:GOTO180
160 IF K$=H$ANDY>0 THEN Y=Y-1:GOTO180
170 GOTO 280
180 PRINT K$;:P%=0:M%=1000
190 FORI=3TO1STEP-1:IF F(I)=1 THEN270
200 A=X-X(I):B=Y-Y(I):D%=SQR(A*A+B*B)
210 IF D%<>0 THEN 250
220 EXPLODE:F(I)=1:N=N+1
230 IF F(1) AND F(2) AND F(3)THEN410
240 GOTO 280
250 SOUND I,80*D%,0:P%=P%+2^(I-1)
```

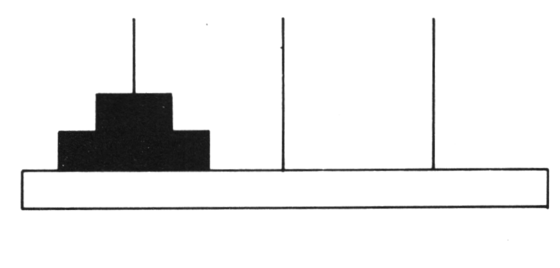
```
260 IF D%<M% THEN M%=D%
270 NEXT:PLAY P%,0,3,100*M%
280 NEXT T
290 :
300 CLS:PAPER 2:INK 0
310 FOR I=1 TO 9:PRINT:NEXT
320 PRINTSPC(4);"Vous avez trouve ";
330 PRINTN;"bidules":PLAY 1,0,7,10
340 MUSIC 1,3,3,15:WAIT 80
350 MUSIC 1,3,5,15:WAIT 80
360 MUSIC 1,3,1,15:WAIT 80
370 MUSIC 1,2,1,15:WAIT 80
380 MUSIC 1,2,8,15:WAIT 80
390 PLAY 0,0,0,0:PRINT CHR$(6):END
400 :
410 FOR I=1 TO 3:F(I)=0:NEXT
420 W=INT(2*W/3)
430 MUSIC 1,3,10,0:PLAY 1,0,3,1500
440 WAIT 400:PLAY 0,0,0,0
450 GOTO 70
```

Attention à ne pas oublier le ; de la ligne 18Ø.

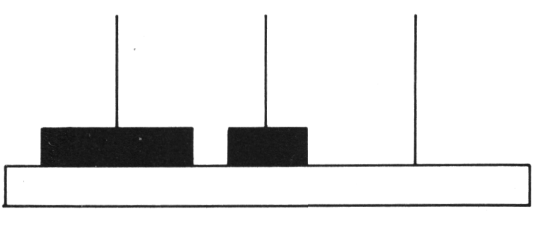
1.5 TOUR DE HANOÏ

Voici maintenant un jeu de réflexion. Le principe en est simple : on dispose d'un support comportant trois tiges verticales et de disques de couleurs et de tailles variées. Ces disques sont percés en leur centre, et peuvent s'enfiler sur les tiges. Au début du jeu, on empile tous les disques sur l'une des tiges, dans l'ordre des tailles décroissantes. Ainsi, le plus grand disque est au-dessous de la pile de disques tandis que le plus petit est au sommet. Le but du jeu est de transférer toute cette pile sur l'une des deux autres tiges. Cela n'est pas si simple, car vous ne pouvez transférer d'une tige à une autre qu'un seul disque à la fois. De plus, à aucun moment, on ne peut avoir un disque enfilé au-dessus d'un plus petit.

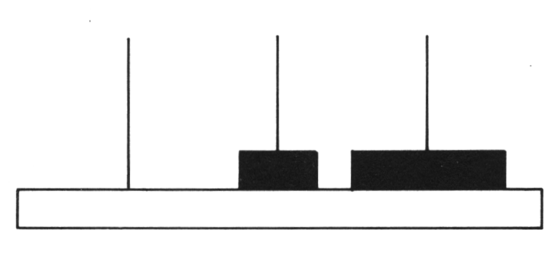
Voici un exemple avec deux disques :



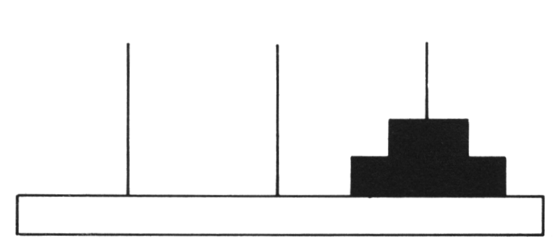
Au départ, on a deux disques sur la tige de gauche. Il faut les amener sur la tige de droite.



1er coup: On déplace le petit disque de la tige de gauche à celle du milieu.



2e coup: On déplace le grand disque de la tige de gauche à celle de droite.



3e coup et dernier coup: On déplace le petit disque de la tige du milieu à celle de droite.

Le programme vous demande tout d'abord:

Nombre de disques?

Tapez 2 puis RETURN. On vous demande alors:

Tige de départ (Ø, 1, 2)?

Tapez Ø puis RETURN. On vous demande encore:

Tige d'arrivée (Ø, 1, 2)?

Tapez 2 puis RETURN. L'écran passe en mode haute résolution et le support est dessiné. Les tiges sont numérotées de Ø à 2. Deux disques sont enfilés sur la tige Ø. On vous demande :

Départ, arrivée (Ø, 1, 2) ?

Tapez, dans l'ordre :

```
Ø,1 return
Ø,2 return
1,2 return
```

Vous pouvez recommencer avec plus de disques, le jeu se complique à partir de 5 disques. Vous pouvez, au début, partir d'une autre tige que celle de gauche, pour arriver sur une autre que celle de droite.

Voici quelques explications sur le fonctionnement de ce programme. Les lignes 1Ø à 12Ø servent à demander le nombre de disques N, le numéro de la tige de départ DE ainsi que celui AR de la tige d'arrivée.

6Ø Les disques sont numérotés de 1 à N, N est celui de diamètre le plus grand. Le tableau TIGE sert à stocker pour chaque tige, les numéros des disques qui y sont enfilés. Le tableau T sert à stocker, pour chaque tige, le nombre de disques qui y sont enfilés.

Les lignes 21Ø à 29Ø servent à tracer le support et placer les disques sur la tige de départ.

23Ø On trace une tige verticale.

24Ø On trace le numéro de la tige. Pour I=1, I+47 vaut 48, ce qui est le code ASCII du chiffre "Ø". De même, 49 est le code ASCII de "1" et 5Ø est celui de "2".

27Ø Le sous-programme 1Ø1Ø empile le disque numéro A2 sur la tige numéro A1. La boucle 26Ø-28Ø empile donc tous les disques sur la tige de départ DE, en commençant par le plus grand.

Le jeu lui-même est constitué des lignes 31Ø à 43Ø.

35Ø Si la tige de départ A1 est vide, s'il n'y a pas de disque, il y a erreur, on va en 38Ø.

360 Si la tige d'arrivée A3 est vide, pas de problème, on peut prendre le dernier disque de la tige A1 pour le mettre sur la tige A3, ce qui est fait en 390.

370 On vérifie que le dernier disque enfilé sur A1, celui que l'on va déplacer, est bien plus petit que le dernier disque enfilé sur A2. Si tel est le cas, on va en 390.

390 Le sous-programme 2010 dépile un disque de la tige A1. Après l'exécution du sous-programme, A2 contient le numéro du disque qui a été dépilé.

400 Le sous-programme 1010 empile le disque numéro A2 sur la tige de numéro A1. Le numéro de la tige d'arrivée, A3, a été placé dans A1 à la ligne précédente.

410 Si il a moins de N disques sur la tige de numéro AR, ce n'est pas terminé, on va donc en 310 pour continuer.

```
10 REM TOUR DE HANOI*****************
20 :
30 TEXT:CLS:PRINT:PRINT
40 INPUT"Nombre de disques ";N
50 IF N<1 OR N>10 THEN PING:GOTO 40
60 DIM TIGE(2,N-1),T(2)
70 INPUT"Tige de depart (0,1,2) ";DE
80 IF DE<0 OR DE>2 THEN PING:GOTO 70
90 INPUT"Tige d'arrivee (0,1,2) ";AR
100 IF AR<0 OR AR>2 THEN PING:GOTO 90
110 IF AR=DE THEN PRINT"Pff !":GOTO70
120 :
210 HIRES
220 FOR I=1 TO 3
230 CURSET80*I-40,199,3:DRAW 0,-150,1
240 CURMOV -2,-15,3:CHAR I+47,0,1
250 NEXT I
260 FOR I=N TO 1 STEP -1
270 A1=DE:A2=I:GOSUB 1010
280 NEXT I
290 :
310 REM DEBUT TRAITEMENT-------------
320 INPUT"Depart,arrivee (0,1,2) ";A1,A3
```

```
330 IF A3<0 OR A3>2 THEN PING:GOTO320
340 IF A1<0 OR A1>2 THEN PING:GOTO320
350 IF T(A1)<=0 THEN 380
360 IF T(A3)<=0 THEN 390
370 IF TIGE(A1,T(A1)-1)<TIGE(A3,T(A3)-1) THEN 39
380 ZAP:WAIT 30:EXPLODE:GOTO 310
390 GOSUB 2010:A1=A3
400 GOSUB 1010
410 IF T(AR)<N THEN 310
420 END
430 :
1010 REM S-PROG EMPILE DISQUE********
1020 L1=190-10*T(A1):C1=80*A1+39
1030 CURSET-78*(A1>0)-84*(A1>1),L1,3
1040 FILL 10,1,A2-7*INT((A2-1)/7)
1050 CURSET72-84*(A1>0)-78*(A1>1),L1,3
1060 FILL 10,1,7
1070 GOSUB 2060
1080 TIGE(A1,T(A1))=A2:T(A1)=T(A1)+1
1090 RETURN
1100 :
2010 REM S-PROG DEPILE DISQUE********
2020 T(A1)=T(A1)-1:A2=TIGE(A1,T(A1))
2030 L1=190-10*T(A1):C1=80*A1+39
2040 CURSET-78*(A1>0)-84*(A1>1),L1,3
2050 FILL 10,1,7
2060 FOR J=0 TO 9
2070 CURSET C1-3*A2,L1+J,3
2080 DRAW 6*A2+1,0,2
2090 NEXT J
2100 RETURN
```

1.6 SOLUTION DE LA TOUR DE HANOÏ

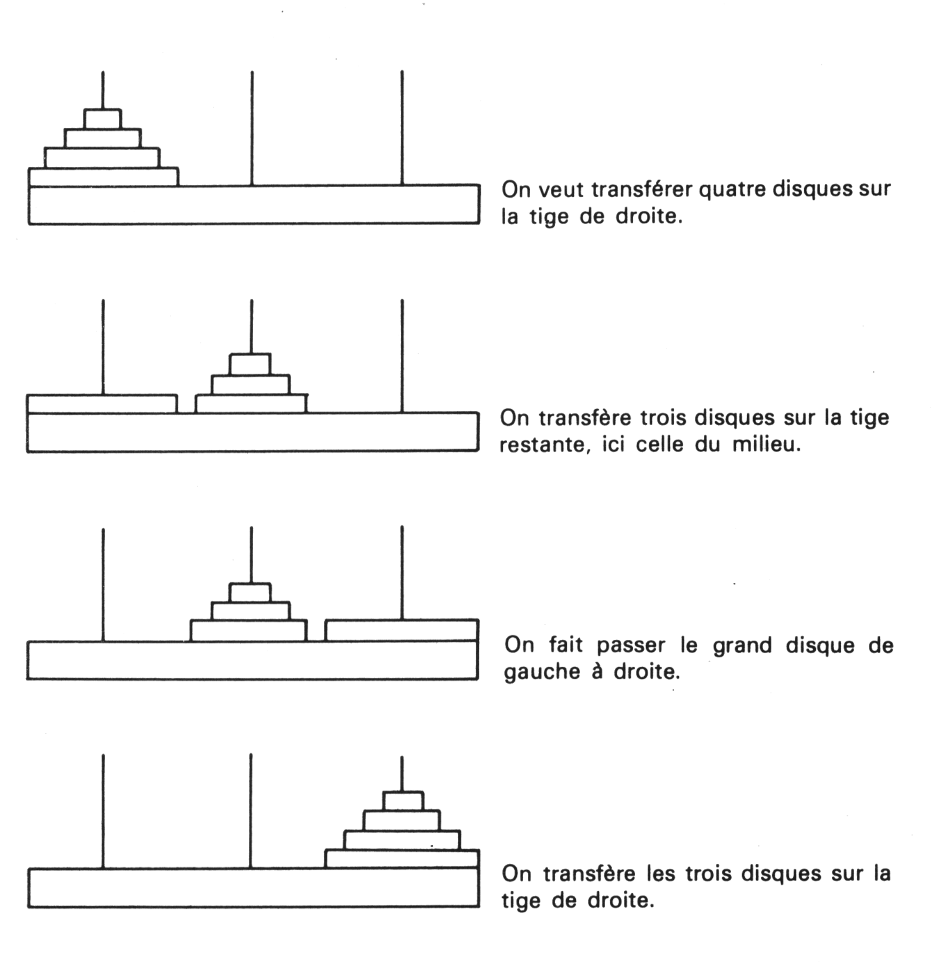
Ce programme, qui résoud le problème de la tour de Hanoï, se trouve dans tous les manuels d'informatique, car il illustre la notion de sous-programme récursif, c'est-à-dire un sous-programme capable de s'appeler lui-même. Il peut être intéressant de voir comment ceci peut être programmé sur ORIC.

Le transfert de N disques d'une tige de départ DE à une tige d'arrivée AR peut se décomposer en trois étapes.

1) On transfère N−1 disques de la tige DE sur la tige restante.

2) Il reste un disque, le plus grand, sur la tige DE. Il n'y a rien sur la tige AR, les autres disques sont empilés sur la troisième tige. On transfère le grand disque de la tige DE à AR.

3) Il ne reste plus qu'à transférer les N−1 disques restants de la tige où ils sont vers la tige AR, et le tour est joué!



On veut transférer quatre disques sur la tige de droite.

On transfère trois disques sur la tige restante, ici celle du milieu.

On fait passer le grand disque de gauche à droite.

On transfère les trois disques sur la tige de droite.

Le programme correspondant peut être décrit comme suit:

```
Programme solution de la tour de Hanoï:
    lire le nombre de disques N
    lire le numéro de la tige de départ DE
    lire le numéro de la tige d'arrivée AR
    exécuter Hanoï (N,DE,AR)
Fin
```

```
Sous-programme Hanoï (B1,B2,B3):
    si B1 vaut 1 alors:
      dépiler le disque du dessus de la tige B2
      empiler ce disque sur la tige B3
    sinon:
      calculer B4=numéro de la tige restante, ni B2, ni B3
      exécuter Hanoï (B1−1, B2, B4)
      exécuter Hanoï (1, B2, B3)
      exécuter Hanoï (B1−1, B4, B3)
Retour
```

Le sous-programme Hanoï est récursif, ce qui signifie que les paramètres locaux B1, B2, B3 et B4 ainsi que l'adresse de retour L doivent être empilées lorsque l'on entre dans le sous-programme. Ce qui est fait en 42Ø-45Ø. Il faut aussi dépiler ces quantités en sortie du sous-programme, ce qui est fait en 56Ø-58Ø.

```
10 REM SOLUTION TOUR DE HANOI********
20 :
30 TEXT:CLS:PRINT:PRINT
40 INPUT"Nombre de disques ";N
50 IF N<1 OR N>10 THEN PING:GOTO 40
60 DIM TIGE(2,N-1),T(2),PYLE(100):PP=0
70 INPUT"Tige de depart (0,1,2) ";DE
80 IF DE<0 OR DE>2 THEN PING:GOTO 70
90 INPUT"Tige d'arrivee (0,1,2) ";AR
100 IF AR<0 OR AR>2 THEN PING:GOTO 90
110 IF AR=DE THEN PRINT"Pff !":GOTO70
120 :
210 HIRES
220 FOR I=1 TO 3
230 CURSET80*I-40,199,3:DRAW 0,-150,1
240 CURMOV -2,-15,3:CHAR I+47,0,1
```

```
250 NEXT I
260 FOR I=N TO 1 STEP -1
270 A1=DE:A2=I:GOSUB 1010
280 NEXT I
290 :
310 REM DEBUT TRAITEMENT-------------
320 B1=N:B2=DE:B3=AR:L=1:GOTO 410
330 END
340 :
410 REM S-PROG HANOI*****************
420 PYLE(PP)=L:PYLE(PP+1)=B1
430 PYLE(PP+2)=B2:PYLE(PP+3)=B3
440 PYLE(PP+4)=B4:PP=PP+5
450 :
460 IF B1<>1 THEN 510
470 A1=B2:GOSUB 2010
480 A1=B3:GOSUB 1010
490 GOTO 560
500 :
510 B4=3-B2-B3
520 B1=B1-1:B3=B4:L=2:GOTO 410
530 B1=1:B3=PYLE(PP-2):L=3:GOTO 410
540 B1=PYLE(PP-4)-1:B2=B4:L=4:GOTO 410
550 :
560 PP=PP-5:B4=PYLE(PP+4)
570 B3=PYLE(PP+3):B2=PYLE(PP+2)
580 B1=PYLE(PP+1):L=PYLE(PP)
590 ON L GOTO 330,530,540,550
600 :
1010 REM S-PROG EMPILE DISQUE********
1020 L1=190-10*T(A1):C1=80*A1+39
1030 CURSET-78*(A1>0)-84*(A1>1),L1,3
1040 FILL 10,1,A2-7*INT((A2-1)/7)
1050 CURSET72-84*(A1>0)-78*(A1>1),L1,3
1060 FILL 10,1,7
1070 GOSUB 2060
1080 TIGE(A1,T(A1))=A2:T(A1)=T(A1)+1
1090 RETURN
1100 :
2010 REM S-PROG DEPILE DISQUE********
2020 T(A1)=T(A1)-1:A2=TIGE(A1,T(A1))
```

```
2030 L1=190-10*T(A1):C1=80*A1+39
2040 CURSET-78*(A1>0)-84*(A1>1),L1,3
2050 FILL 10,1,7
2060 FOR J=0 TO 9
2070 CURSET C1-3*A2,L1+J,3
2080 DRAW 6*A2+1,0,2
2090 NEXT J
2100 RETURN
```

1.7 CHUTES D'ÉTOILES

Après ces deux programmes sur la tour de Hanoï, nous entrons maintenant dans le monde des jeux intersidéraux. Vous êtes maintenant le commandant d'un vaisseau intersidéral. Vous traversez un champ d'étoiles énergétiques, il faut essayer d'avaler ces particules en plaçant votre vaisseau devant, vous disposez des deux touches "flèche à gauche" et "flèche à droite" pour vous diriger. Attention : certaines de ces étoiles clignotent, elles sont en fin de leur existence et risquent d'exploser. Il convient donc de les éviter. Une particule rouge vous rapporte un point, une verte deux points, une jaune trois points, une bleue quatre points, une mauve cinq points et une bleu ciel vaut six points. Et maintenant, bonne chance !

Ce jeu utilise le mode LORES 1, qui rappelons-le, est un mode TEXT, l'ordre LORES 1 place simplement un caractère CHR$(9) au début de chacune des 27 lignes de l'écran. Ce caractère impose au système vidéo de l'ORIC l'utilisation du jeu de caractères semi-graphiques. Chaque étoile est un caractère semi-graphique de code ASCII 35, le vaisseau est lui constitué des caractères de codes 33 et 34.

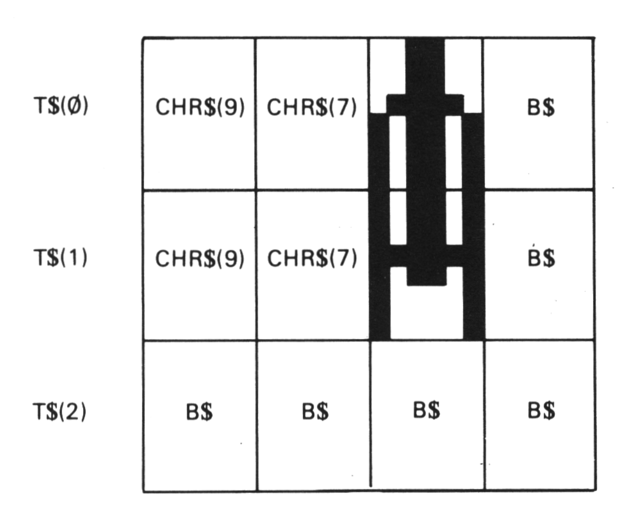
Le début du programme, de 1Ø à 16Ø, sert à redéfinir les caractères semi-graphiques employés, ainsi que quelques constantes.

4Ø B% est le code ASCII du "blanc", T% et Q% sont les codes ASCII de la tête et de la queue du vaisseau, alors que A% est celui de l'étoile.

8Ø-9Ø X% et Y% sont les coordonnées sur l'écran du vaisseau. CHR$(17) et CHR$(2Ø) suppriment le curseur et le mot CAPS.

1ØØ G$ et D$ contiennent les codes ASCII des touches "flèche à gauche" et "flèche à droite". H$ contient celui de "flèche en haut".

11Ø à 14Ø Le vaisseau est obtenu en affichant trois chaînes T$(Ø), T$(1) et T$(2).



Les CHR$(9) forcent l'utilisation des caractères semi-graphiques non clignotants, ainsi le vaisseau ne clignotera pas lorsqu'une étoile clignotante se trouvera à sa gauche. Le CHR$(7) détermine la couleur du vaisseau, blanc.

Lorsque l'on déplace le vaisseau à gauche, les deux blancs **B$** situés à droite effacent le vaisseau précédent. Lorsque l'on se déplace à droite, le vaisseau précédent est remplacé, effacé par les deux CHR$(7). Enfin les quatre blancs de la chaîne **T$**(2) effacent le bas du vaisseau précédent après chaque déplacement vers le bas du champ d'étoiles.

15Ø S est l'adresse-mémoire de l'octet qui correspond au 2[e] caractère de gauche de la première ligne de l'écran, elle servira à la ligne 22Ø. Les constantes L et M seront aussi utilisées à la ligne 22Ø.

La boucle de jeu s'étend de 17Ø à 3ØØ.

19Ø On tire au hasard un nombre Z% compris entre 2 et 38.

2ØØ En Z% on place un caractère dont le code ASCII est aléatoire et compris entre 1 et 6. C'est un code de couleur, qui détermine la teinte

des caractères situés après lui sur la ligne. En Z%+1, on met une étoile, qui a la couleur définie par le caractère précédent.

21Ø On appelle un sous-programme en langage machine qui déplace l'écran d'une ligne vers le bas.

22Ø On place soit un CHR$(9) soit un CHR$(13) au début de la première ligne de l'écran. L'étoile qui a été placé sur cette même ligne écran en 2ØØ sera donc clignotante avec CHR$(13) et fixe avec CHR$(9).

23Ø Lit le clavier.

24Ø Si la touche frappée est "flèche à gauche" et si X% est supérieur à Ø l'expression (C$=G$ AND X%>Ø) est vraie et vaut −1, on décrémente X%. Si on a frappé "flèche à droite", c'est (C$=D$ AND X%<36) qui vaut −1 et on incrémente X%.

25Ø On trace le vaisseau.

26Ø On regarde si l'on est en face d'une étoile.

27Ø On regarde si l'étoile clignote.

28Ø SCRN(X%+1,Y%−1) donne le code ASCII du code de couleur de l'étoile, c'est un nombre compris entre 1 et 6 que l'on ajoute au score U%.

A la ligne 3Ø, on réserve 6Ø octets pour placer le sous-programme machine, ce qui est fait en 1ØØØ.

1Ø1Ø En #A6-#A7 se trouve la valeur de HIMEM, que l'on range en A.

1Ø2Ø-1Ø3Ø On place 55 octets en mémoire à partir de A.

1Ø4Ø Le sous-programme situé en A appelle lui-même un sous-programme débutant en A+43. PL contient donc l'adresse de ce second sous-programme. En A+2Ø, on trouve une instruction JSR vers ce second sous-programme. On place donc en A+21 l'adresse de ce second sous-programme.

* Décalage vers le bas de l'écran en mode TEXT ou LORES :

A21A	A	LDX	#26	Initialise compteur de lignes.
A9B8		LDA	#$B8	L'adresse mémoire $BFB8 du
85ØØ		STA	Ø	premier octet de la dernière ligne
A9BF		LDA	#$BF	de l'écran TEXT est rangée en
85Ø1		STA	1	Ø-1.
AØ27	Tligne	LDY	#39	Initialise compteur de caractères.

A5ØØ		LDA	Ø	L'adresse rangée en (Ø-1) est recopiée.
85Ø2		STA	2	
A5Ø1		LDA	1	en (2-3)
85Ø3		STA	3	puis
2ØXXXX		JSR	PL	on retranche 4Ø à (Ø-1).
B1ØØ	Trcara	LDA	(Ø), Y	Prendre un octet et
91Ø2		LDA	(2), Y	le recopier à la ligne du dessous.
88		DEY		décrémenter compteur de caractères
1ØF9		BPL	Trcara	Si pas nul, continuer la ligne.
CA		DEX		Sinon, décrémenter compteur de lignes.
1ØE9		BPL	Tligne	Si pas nul, passer à la ligne suivante.
AØ27	Ligne Ø	LDY	#39	La 1re ligne de l'écran va être remplie de blancs (code 32).
A92Ø		LDA	#32	
91ØØ	boucle	STA	(Ø), Y	
CA		DEY		
1ØFB		BPL	boucle	
6Ø		RTS		

* Oter 4Ø, nombre d'octet par ligne, à (Ø-1) :

A5ØØ	PL	LDA	Ø	Prendre l'octet de poids faible.
38		SEC		Préparer la soustraction.
E928		SBC	#4Ø	Oter 4Ø.
85ØØ		STA	Ø	Ranger l'octet de poids faible.
BØØ2		BCS	Retour	Si retenue, retour.
C6Ø1		DEC	Ø1	Sinon décrémenter octet poids fort.
6Ø	Retour	RTS		

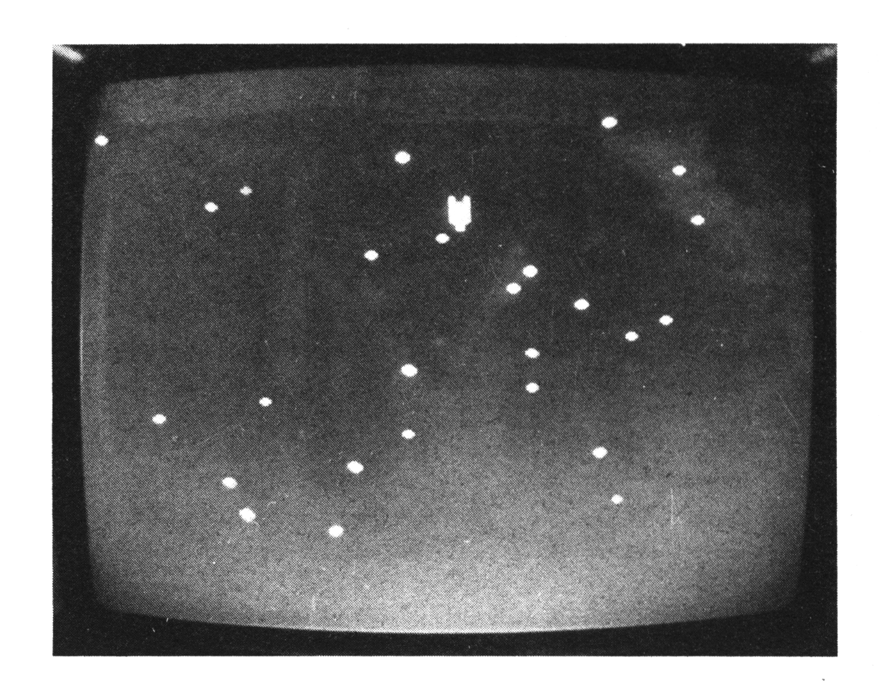
```
10 REM ETOILES************************
20 :
30 HIMEM £9800-50:GOSUB 1000
40 B%=32:T%=33:Q%=34:A%=35
50 PAPER 0:INK 0:LORES 1:S=£B800+8*T%
60 S=£B800+8*T%
70 GOSUB 2000
80 X%=15:Y%=20
90 PRINT CHR$(17);CHR$(20)
100 G$=CHR$(8):D$=CHR$(9)
110 DIM T$(2):B$=CHR$(B%)
```

```
120 T$(0)=CHR$(9)+CHR$(7)+CHR$(T%)+B$
130 T$(1)=CHR$(9)+CHR$(7)+CHR$(Q%)+B$
140 T$(2)=B$+B$+B$+B$
150 S=£BB80+41:L=9:M=13
160 :
170 REM DEBUT DU PROGRAMME-----------
180 FOR T=1 TO 400
190 Z%=37*RND(1)+2
200 PLOTZ%,0,1+6*RND(1):PLOTZ%+1,0,A%
210 CALL A
220 POKE S,L:IF RND(1)<.5THEN POKES,M
230 C$=KEY$
240 X%=X%+(C$=G$ANDX%>0)-(C$=D$ANDX%<36)
250 FOR I=0TO2:PLOTX%,Y%+I,T$(I):NEXT
260 IF SCRN(X%+2,Y%-1)<>A% THEN 290
270 IF SCRN(1,Y%-1)=13 THEN 330
280 ZAP:U%=U%+SCRN(X%+1,Y%-1)
290 Z=FRE("")
300 NEXT T:WAIT 200:GOTO 350
310 :
320 REM FIN DU PROGRAMME-------------
330 FORI=0TO2:PLOT X%+1,Y%+I,1:NEXT
340 EXPLODE:WAIT 300
350 TEXT:PAPER 2:INK 0:CLS
360 PRINT:PRINT:PRINT:PRINT:PRINT
370 PRINT SPC(10);"Vous avez gagne"
380 PRINT:PRINT:PRINT
390 PRINT SPC(15);CHR$(27);"A";
400 PRINT CHR$(27);"N";U%
410 PRINT SPC(15);CHR$(27);"A";
420 PRINT CHR$(27);"N";U%:PRINT
430 PRINT:PRINT:PRINTSPC(15);"points"
440 :
450 B=INT(U%/30):PRINT:PRINT:PRINT
460 IF B=0 THEN SHOOT:GOTO 500
470 IF B>5 THEN EXPLODE:GOTO 560
480 ON B GOTO 510,520,530,540,550
490 :
500 PRINT"Vous etes nul.":GOTO 580
510 PRINT"Pas terrible.":GOTO 580
520 PRINT"Bof ...":GOTO 580
```

```
530 PRINT"Pas mal.":GOTO 580
540 PRINT"Felicitations":GOTO 580
550 PRINT"Quel score !!":GOTO 580
560 PRINT"Fantastique ! Formidable !"
570 :
580 PRINT CHR$(17);CHR$(20):END
590 :
1000 REM SOUS PROGRAMME MACHINE******
1010 A=DEEK(£A6)
1020 FOR I=A TO A+54:READ V
1030 POKE I,V:NEXT
1040 PL=A+43:DOKE A+21,PL
1050 RETURN
1060 DATA£A2,£1A,£A9,£B8,£85,£00,£A9
1070 DATA£BF,£85,£01,£A0,£27,£A5,£00
1080 DATA£85,£02,£A5,£01,£85,£03,£20
1090 DATA£FF,£FF,£B1,£00,£91,£02,£88
1100 DATA£10,£F9,£CA,£10,£E9,£A0,£27
1110 DATA£A9,£20,£91,£00,£CA,£10,£FB
1120 DATA£60
1130 DATA£A5,£00,£38,£E9,£28,£85,£00
1140 DATA£B0,£02,£C6,£01,£60
1150 :
2000 REM CARACTERES******************
2010 FORI=STOS+23:READV:POKEI,V:NEXT
2020 DATA 12,12,12,30,45,45,45,45
2030 DATA 45,45,45,63,45,33,33,33
2040 DATA 0,0,12,30,30,12,0,0
2050 RETURN
```

Note: le caractère "£" est à remplacer par "#".

Ne pas oublier les ; des lignes 39Ø et 41Ø.



1.8 VOYAGE DANS L'ESPACE

Après CHUTE D'ÉTOILES, nous poursuivons notre voyage dans l'espace. Le vide intersidéral est décidément très peuplé, car vous traversez de nouveau un champ d'étoiles. Mais attention : cette fois-ci, ce sont des étoiles très dures, qu'il faut absolument éviter. Vous disposez toujours des flèches à gauche et droite pour manœuvrer. Au départ, vous avez un crédit de 9, ce qui signifie que vous pouvez percuter neuf étoiles au maximum. Il ne me reste plus qu'à vous souhaiter bon voyage !

3Ø On efface l'écran, on efface le curseur, le mot CAPS, on supprime le "plop" produit par les touches du clavier;

4Ø Bien qu'il ne soit plus affiché, le curseur existe toujours dans l'ORIC, il indique l'endroit où sera affiché le prochain message. L'écran comporte 27 lignes, on envoie le curseur sur la dernière ligne de l'écran.

5Ø S est l'adresse-mémoire de la première ligne de l'écan. (On ne compte pas la ligne où se trouve CAPS).

6Ø H%=42 est le code ASCII de l'étoile. G$ et D$ contiennent les codes ASCII des touches "flèche à gauche" et "flèche à droite".

8Ø On place trois étoiles sur la dernière ligne de l'écran.

9Ø Comme le curseur est en bas de l'écran, PRINT décale tout le contenu de l'écran vers le haut. Puis on lit le clavier.

1ØØ Ceci, comme dans le jeu précédent, incrémente ou décrémente X selon la touche frappée.

11Ø à 13Ø On regarde si l'on a rencontré une étoile. Si oui, on décrémente le crédit V.

14Ø On fabrique le vaisseau. V est un nombre compris entre 1 et 9. V+48 est compris entre 49 et 57, ce qui correspond aux codes ASCII des caractères "1" à "9".

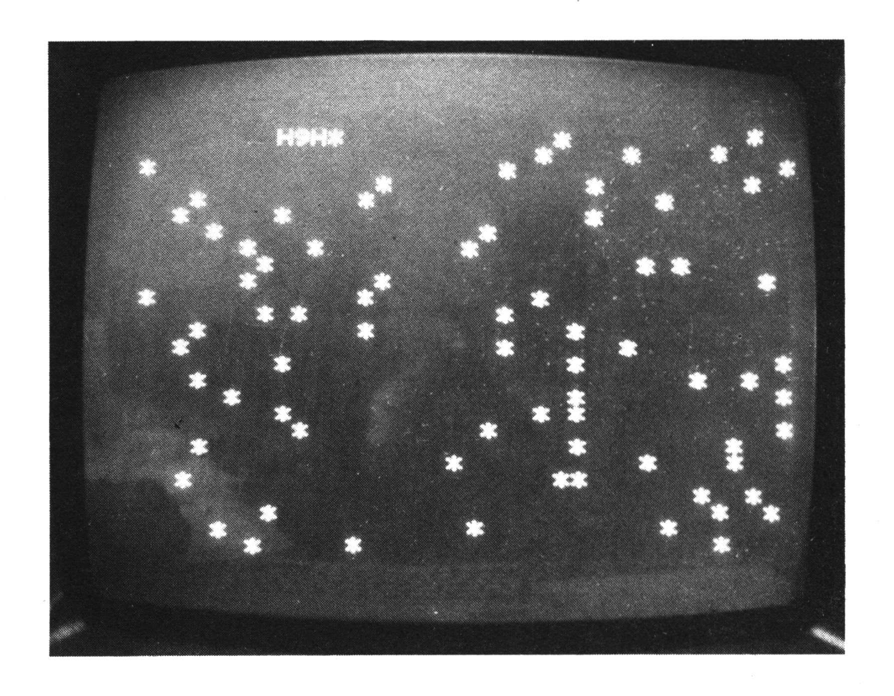
15Ø On affiche le vaisseau et on incrémente le score.

16Ø On modifie la couleur de façon aléatoire.

21Ø Le caractère CHR$(2) donne au nombre la couleur verte. Pour que les mots "année lumière..." qui sont à la suite ne soient pas verts eux aussi, on affiche CHR$(3), ces mots seront donc jaunes.

```
10 REM VOYAGE DANS L'ESPACE**********
20 :
30 CLS:PRINTCHR$(17);CHR$(20);CHR$(6)
40 FOR I=1 TO 27:PRINT:NEXT:PAPER 0
50 X=14:Y=26:V=9:S=£BBA8:L=38
60 N%=0:H%=42:G$=CHR$(8):D$=CHR$(9)
70 :
80 FORI=1TO3:PLOT2+RND(1)*L,Y,H%:NEXT
90 PRINT:K$=KEY$
100 X=X+(K$=G$ANDX>1)-(K$=D$ANDX<36)
110 FOR I=0 TO 2
120 IF PEEK(S+X+I)=H% THEN V=V-1:PING
130 NEXT:IF V<1 THEN 180
140 V$="H"+CHR$(V+48)+"H"
150 PLOT X,0,V$:N%=N%+1
160 INK 1+7*RND(1):GOTO 80
```

```
170 :
180 EXPLODE:WAIT 200:PAPER 0:INK 3
190 PLOT 9,13,"Vous avez voyage dans"
200 PLOT 11,15,"l'espace pendant"
210 PLOT 8,17,CHR$(2)+STR$(N%)+CHR$(3)
220 PLOT 13,17,"annees lumiere..."
230 PRINT CHR$(17);CHR$(20);CHR$(6)
```



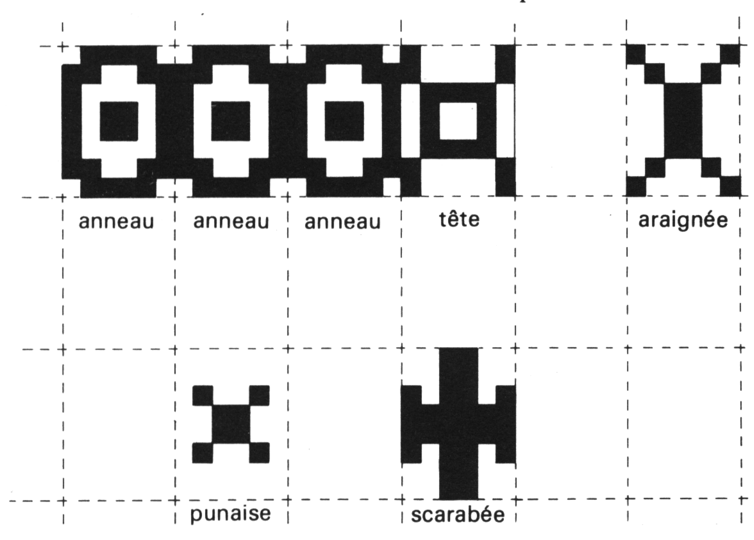
1.9 CHENILLE

Voici maintenant un "classique" des jeux sur micro-ordinateurs : Le jeu de la chenille. Une pauvre petite chenille affamée se promène sur votre écran, cherchant sa nourriture. Elle mange des punaises, des araignées et des gros scarabées. Votre rôle consiste à guider la petite chenille avec les quatre touches fléchées. A chaque fois qu'elle mange une punaise, elle grandit d'un anneau. Une araignée la fait grandir de trois anneaux, et un scarabée lui donne dix anneaux de plus. Le but du jeu est de faire grandir le plus possible la petite chenille.

Attention : cet animal est très vorace, il avale vraiment tout ce qui se présente à ses mâchoires. Il ne faut pas que sa tête passe sur son corps, car il se dévore immédiatement et en meurt. De plus, les gros scarabées sont très lourds à digérer, la chenille ne peut en manger qu'après avoir avalé une araignée, qui contient des sucs digestifs.

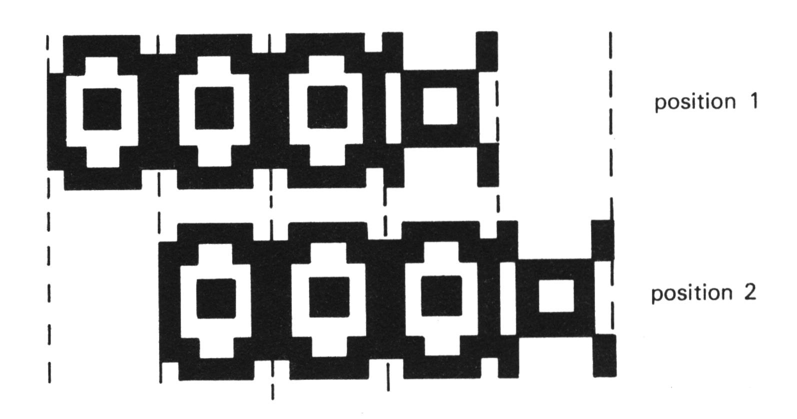
La difficulté du jeu commence à se manifester lorsque l'on atteint une longueur d'environ deux cents anneaux : Il faut soigneusement choisir son chemin pour éviter d'être pris dans les replis du corps qui occupe une bonne partie de l'écran. Pour vous aider, le nombre de scarabées que vous pouvez avaler sans risquer l'indigestion est inscrit en rouge en bas de l'écran.

La tête de la chenille, chacun de ses anneaux, la punaise, l'araignée et le scarabée sont bien sûr des caractères que l'on a redéfini.



La difficulté de ce programme réside dans le déplacement de la chenille. On ne peut pas déplacer sur l'écran toute la chenille lorsque vous frappez une touche, car cela serait beaucoup trop long. Prenons un exemple pour montrer comment l'on procède :

On veut déplacer vers la droite la chenille :



On pourrait bien sûr déplacer vers la droite les quatre caractères qui représentent la chenille. En fait, partant de la position 1, il suffit de :

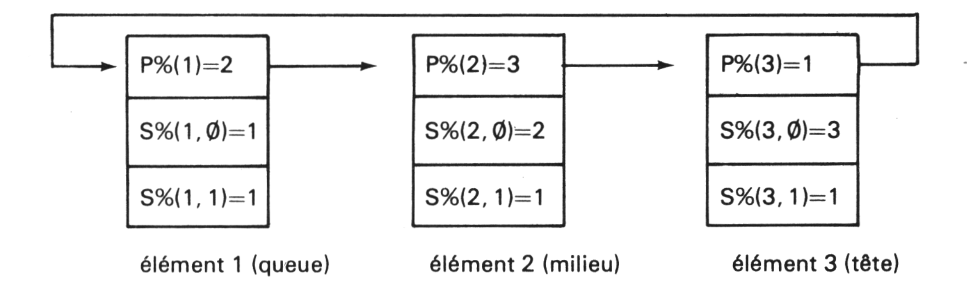
— déplacer la tête vers la droite,

— mettre un anneau à l'endroit où se trouvait la tête avant son déplacement,

— effacer le dernier anneau.

On arrive ainsi à la position 2 en trois étapes, et ceci quelle que soit la longueur de la chenille.

Dans la mémoire de l'ORIC, chaque élément I, anneau ou tête, de la chenille, est représenté par trois nombres : son abscisse sur l'écran S%(I, Ø), son ordonnée S%(I, 1) et un pointeur P%(I). Le pointeur P%(I) contient le numéro de l'élément précédent dans la chenille. Au début du jeu, on a :

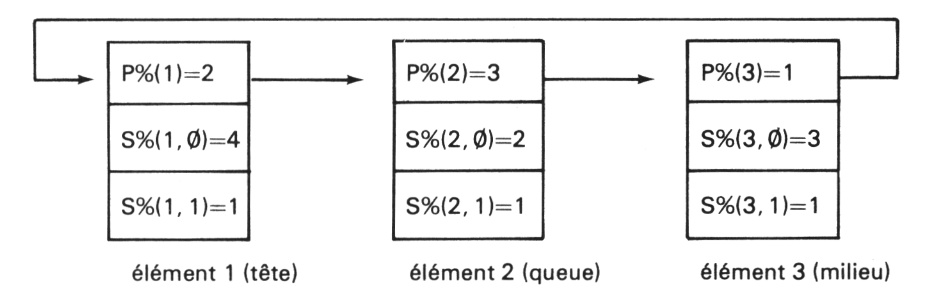


élément 1 (queue) élément 2 (milieu) élément 3 (tête)

On a de plus deux variables TÊTE et QUEUE qui contiennent le numéro de l'élément qui est la tête et celui de l'élément en queue respectivement. Au début du jeu, TÊTE=3 et QUEUE=1. Si l'on déplace la chenille vers la droite :

— On fait progresser en suivant le chaînage les contenus de TÊTE et QUEUE en exécutant TÊTE=P%(TÊTE) et QUEUE=P%(QUEUE).

— On place les nouvelles coordonnées de la tête dans l'élément de numéro TÊTE.

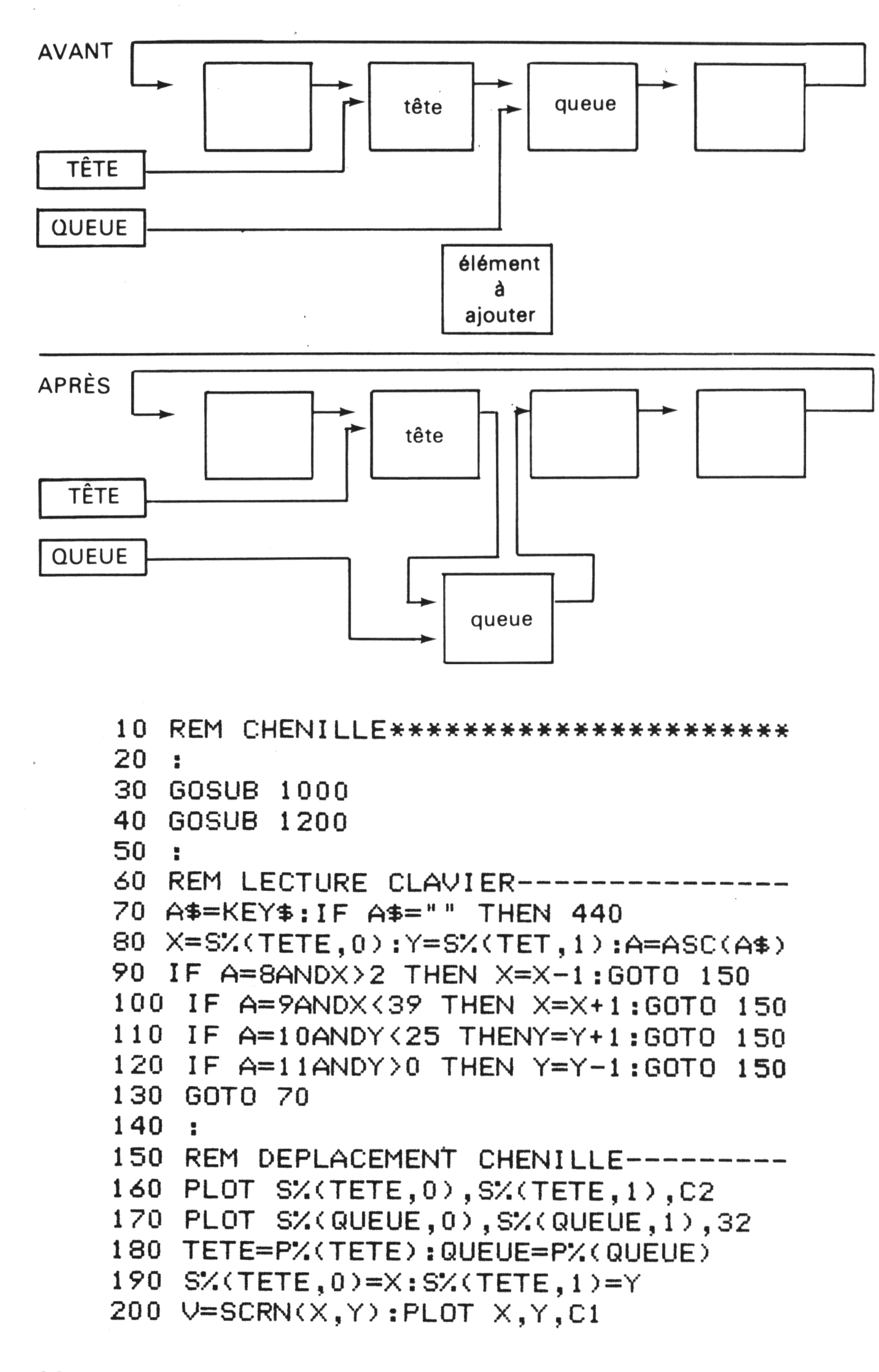


élément 1 (tête) élément 2 (queue) élément 3 (milieu)

TÊTE vaut 1 et QUEUE vaut 2.

Pour déplacer la chenille, on peut seulement modifier le contenu de S%(1, Ø), S%(1, 1), TÊTE et QUEUE. De façon plus générale, n'importe quel déplacement ne modifie que quatre quantités, TÊTE, QUEUE, S%(TÊTE, Ø) et S%(QUEUE, 1), quelle que soit la taille de la chenille.

Pour allonger la chenille d'un élément, on incrémente la variable L, qui contient la longueur de l'animal, puis on ajoute l'élément L en queue.



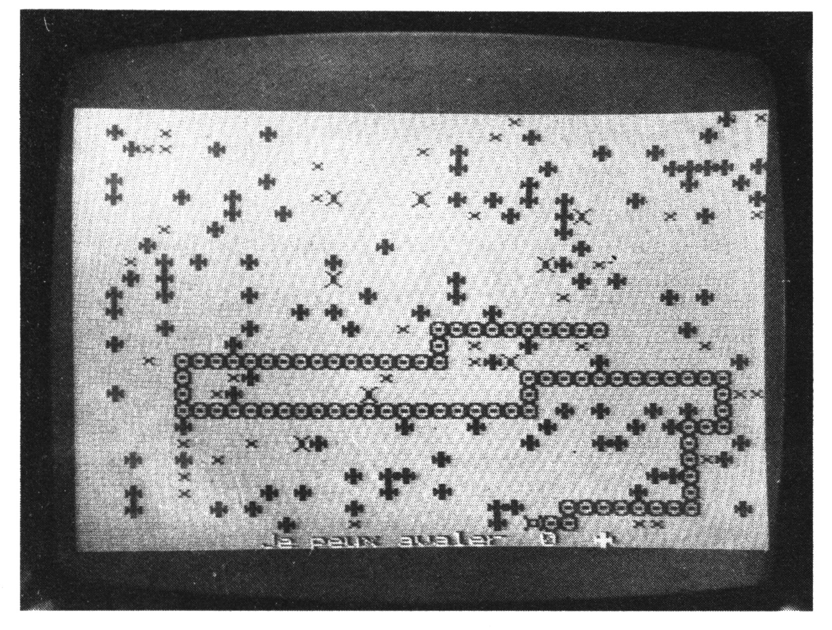

```
10 REM CHENILLE************************
20 :
30 GOSUB 1000
40 GOSUB 1200
50 :
60 REM LECTURE CLAVIER---------------
70 A$=KEY$:IF A$="" THEN 440
80 X=S%(TETE,0):Y=S%(TET,1):A=ASC(A$)
90 IF A=8ANDX>2 THEN X=X-1:GOTO 150
100 IF A=9ANDX<39 THEN X=X+1:GOTO 150
110 IF A=10ANDY<25 THENY=Y+1:GOTO 150
120 IF A=11ANDY>0 THEN Y=Y-1:GOTO 150
130 GOTO 70
140 :
150 REM DEPLACEMENT CHENILLE---------
160 PLOT S%(TETE,0),S%(TETE,1),C2
170 PLOT S%(QUEUE,0),S%(QUEUE,1),32
180 TETE=P%(TETE):QUEUE=P%(QUEUE)
190 S%(TETE,0)=X:S%(TETE,1)=Y
200 V=SCRN(X,Y):PLOT X,Y,C1
```

```
210 :
220 IF V>128 THEN V=V-128
230 IF V=C1 OR V= C2 THEN 550
240 IF V<>C4 THEN 280
250 REM PUNAISE-----------------------
260 ZAP:LL=1:GOSUB 650:GOTO 440
270 :
280 IF V<>C3 THEN 340
290 REM ARAIGNEE----------------------
300 ZAP:LL=3:GOSUB 650::F=F+1
310 MUSIC 1,5,1,0:PLAY 1,1,3,200
320 GOTO 390
330 :
340 IF V<> C5 THEN 440
350 REM GROS SCARABEE-----------------
360 F=F-1:IF F<0 THEN 550
370 MUSIC 1,3,8,0:PLAY 1,1,1,5000
380 LL=10:GOSUB 650
390 PLOT 10,26,"Je peux avaler"
400 PLOT 25,26,STR$(F):PLOT 25,26,32
410 PLOT 29,26,C5
420 :
430 REM GENERATION INSECTES-----------
440 IF RND(1)>.08 THEN 70
450 V=INT(51*RND(1))
460 X=2+38*RND(1):Y=26*RND(1)
470 IF SCRN(X,Y)<>32 THEN 70
480 V=INT(51*RND(1))
490 IF V>46 THEN C=C3
500 IF V>10 AND V<47 THEN C=C5
510 IF V<11 THEN C=C4
520 PLOT X,Y,C:GOTO 70
530 :
540 REM PERDU-------------------------
550 EXPLODE:PLOT16,13,"Score:"
560 L$=STR$(L)
570 L$=RIGHT$(L$,LEN(L$)-1)
580 PLOT 22,13,L$:PRINT CHR$(6);
590 PLOT 10,15,"On rejoue (O/N) ?"
600 GET A$:IF A$="O" THEN 40
610 IF A$<>"N" THEN PING:GOTO 600
```

```
620 CLS:CALL £F8D0:END
630 :
640 REM ALLONGER LA CHENILLE
650 FOR J=1 TO LL:L=L+1
660 P%(L)=QUEUE:P%(TETE)=L
670 S%(L,0)=S%(QUEUE,0)
680 S%(L,1)=S%(QUEUE,1)
690 QUEUE=L
700 NEXT J
710 RETURN
720 :
1000 REM DEFINITION CARACTERES*******
1010 DATA 33,33,30,18,18,30,33,33
1020 DATA 30,51,33,45,45,33,51,30
1030 DATA 33,18,12,12,12,12,18,33
1040 DATA 0,0,18,12,12,18,0,0
1050 DATA 12,12,45,63,63,45,12,12
1060 S=46080+122*8
1070 FOR I=STOS+39:READV:POKEI,V:NEXT
1080 C1=122:C2=123:C3=124:C4=125
1090 C5=126:DIM S%(600,1),P%(600)
1100 RETURN
1110 :
1200 REM INITIALISATIONS*************
1210 TEXT:PAPER 2:INK 0:CLS
1220 L=3:F=0:PRINT CHR$(6);
1230 FOR I=1 TO L
1240 S%(I,0)=I+1:S%(I,1)=1
1250 P%(I)=I+1:PLOT I+1,1,C2:NEXT I
1260 P%(L)=1:TETE=L:QUEUE=1
1270 PLOT L+1,1,C1:PLOT 1,26,1
1280 PLOT 10,26,"Je peux avaler"
1290 PLOT 25,26,48:PLOT 29,26,C5
1300 RETURN
```

Note: le caractère "£" est à remplacer par "#".

L'ordre CALL #F8DØ de la ligne 62Ø recharge les deux jeux de caractères de l'ORIC/ATMOS.



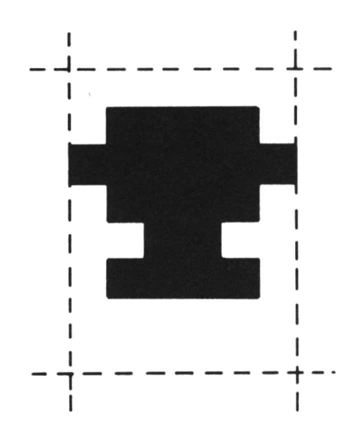
1.10 ENCLUMES

A la suite d'une dispute avec l'un de vos voisins, vous décidez de lancer sur sa maison des projectiles à l'aide d'une catapulte. Mais que lancer ? Il faut en effet des objets assez lourds, capables de produire en tombant suffisamment de dégâts. Votre choix se porte sur un stock d'enclumes, acheté d'occasion.

Avant chaque tir, il faut régler la catapulte, pour fixer d'une part la vitesse de départ de l'enclume et d'autre part l'angle de départ par rapport à l'horizontale. Le tir est compliqué par la présence d'une montagne et par du vent, qui dévie la trajectoire.

A chaque tir, il faut donner la vitesse initiale exprimée en mètres/-seconde, l'angle de départ en degrés. La vitesse du vent est positive s'il va vers la droite de l'écran, négative s'il va vers la gauche.

Voici quelques explications sur le fonctionnement de ce programme. Les lignes 3Ø à 6Ø servent à définir le caractère qui représentera l'enclume. 46592 vaut 46Ø8Ø+8*64, ce qui signifie que l'on a redéfini le caractère de code ASCII 64.



Les lignes 80 à 190 servent à tracer la montagne de pente H. De 110 à 140 on trace le versant gauche tandis que le versant droit est dessiné de 150 à 180. A la ligne 180, on place en bas à droite de l'écran un caractère CHR$(17) sur neuf lignes. Ce caractère donne à l'écran la couleur rouge.

Le bloc 200 à 320 sert à initialiser la position (X, Y) du projectile ainsi que sa vitesse (VX, VY). On tire aussi au hasard la vitesse V du vent. G=9.81 est l'accélération de la pesanteur.

La trajectoire est calculée point par point, en utilisant une méthode d'intégration numérique d'ordre deux. Le projectile est soumis à l'accélération de la pesanteur et à une traînée aérodynamique proportionnelle au carré de sa vitesse V1 par rapport à l'air.

De 330 à 390 on calcule un point.

340 V1 est la vitesse de l'enclume par rapport à l'air, tandis que G1 est la décélération due au frottement aérodynamique.

350 DT est le pas de calcul.

360 On calcule les composantes GX et GY de l'accélération du corps.

370 On calcule les composantes VX et VY de la vitesse.

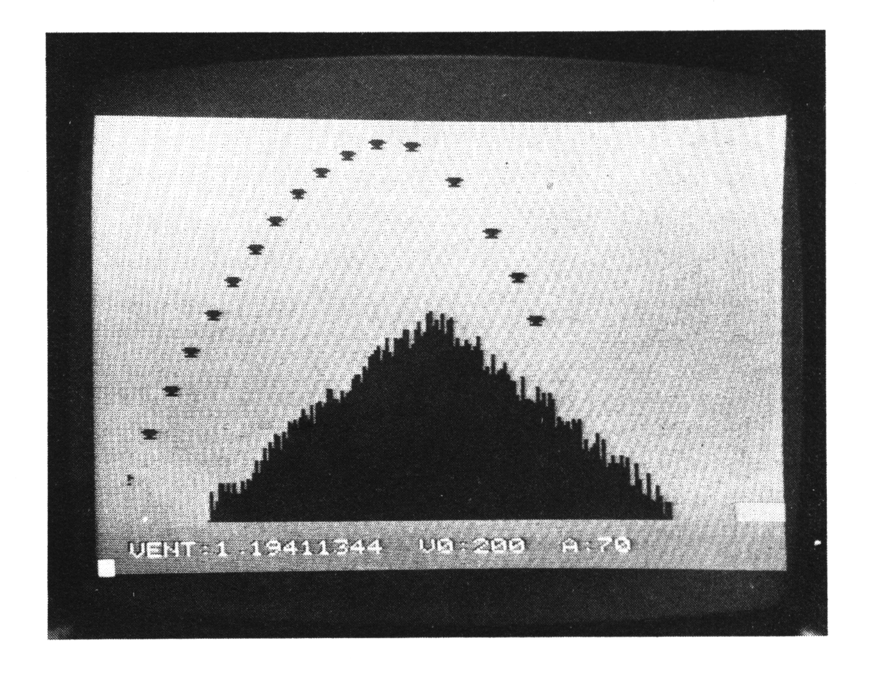
380 On obtient alors la nouvelle position (X, Y)

Les lignes 400 à 500 affichent l'enclume, en vérifiant qu'elle n'est pas sortie de l'écran et qu'elle n'a pas percuté la montagne.

Enfin, les lignes 51Ø à 55Ø terminent le programme.

```
10 REM JEU D'ENCLUMES****************
20 :
30 REM DEFINITION FORME ENCLUME------
40 DATA 0,30,63,30,12,30,0,0
50 TEXT:FOR I=46592 TO 46599
60 READ V:POKE I,V:NEXT
70 :
80 REM TRACE MONTAGNE----------------
90 PAPER 2:INK 4:H=RND(1)*1.5+.2
100 HIRES:PAPER 2:INK 0
110 FOR I=40 TO 119
120 J=INT((I-40)*H+15*RND(1))+1
130 CURSET I,199,3:DRAW 0,-J,1
140 NEXT I
150 FOR I=120 TO 199
160 J=INT((199-I)*H+15*RND(1))+1
170 CURSET I,199,3:DRAW 0,-J,1
180 NEXT I:CURSET222,190,3:FILL9,1,17
190 :
200 REM INITIALISATIONS--------------
210 X=0:Y=0:K=.005:G=9.81
220 V=(RND(1)-.5)*5
230 PRINT"VENT:";V
240 PRINT"DONNER LA VITESSE INITIALE"
250 INPUT V0
260 PRINT"VENT:";V;" V0:";V0
270 PRINT"DONNER L'ANGLE DE DEPART"
280 INPUT A:PRINT
290 PRINT"VENT:";V;" V0:";V0;" A:";A
300 A=A/180*PI
310 VX=V0*COS(A):VY=V0*SIN(A)
320 :
330 REM CALCUL TRAJECTOIRE-----------
340 V1=SQR((VX-V)^2+VY^2):G1=K*V1^2
350 DT=15/V1+.1:D2=DT^2/2
360 GX=-(VX-V)/V1*G1:GY=-VY/V1*G1-G
370 VX=VX+GX*DT:VY=VY+GY*DT
380 X=X+VX*DT+GX*D2:Y=Y+VY*DT+GY*D2
```

```
390 :
400 REM AFFICHAGE--------------------
410 Z=199-Y:IF Z<192 THEN 440
420 IF X>220 AND X<233 THEN 510
430 PING:WAIT 200:GOTO 100
440 IF X<6 OR X>233 OR Z<0 THEN 340
450 IF POINT(X,Z)=0 THEN 470
460 EXPLODE:WAIT 300:GOTO 100
470 CURSET X,Z,0:CHAR 64,0,1
480 SOUND 1,10000/(200-Z),0
490 PLAY 1,0,4,100:GOTO 340
500 :
510 REM COUP AU BUT------------------
520 PAPER 1:CURSET X,192,3
530 CHAR 64,0,1:ZAP:WAIT 40:EXPLODE
540 PRINT:PRINTSPC(27);"Sprotch !!"
550 WAIT 400:SHOOT
```



1.11 MUR DE BRIQUES

Chapeau bas. J'ai l'honneur et l'avantage de vous présenter l'un des ancêtres, l'un des pionniers des jeux vidéos, le jeu de mur de briques. En face de vous se trouve un mur de 4ØØ briques. Vous avez une raquette, et vous devez sans cesse renvoyer une petite balle qui casse les briques sur lesquelles elle rebondit. Le but est bien sûr de casser les 4ØØ briques.

Les touches "flèche à gauche" et "flèche à droite" déplacent la raquette, tandis que "flèche en bas" incline cette raquette vers la gauche et "flèche en haut" l'incline vers la droite. Ainsi, la raquette peut avoir cinq inclinaisons différentes, et l'on peut renvoyer la balle dans une autre direction.

La tactique consiste à percer le mur de part en part, puis d'envoyer la balle dans le trou, pour qu'elle arrive entre le haut du mur et le haut de l'écran. Elle rebondit alors sur le haut de l'écran et sur le mur, en détruisant une brique à chaque fois.

Ce jeu utilise le mode haute résolution HIRES. Les principales variables utilisées sont :

— X, Y : position de la balle sur l'écran.

— VX, VY : vitesse de la balle.

— XR, YR : position de la raquette.

R : angle de la raquette.

— MG, MD, MB et MH sont les marges gauche, droite, basse et haute de l'écran.

— A$, B$, C$ et D$ contiennent les codes ASCII des touches flèches.

Le fonctionnement du programme est le suivant :

3Ø Les initialisations sont laissées à la responsabilité du sous-programme 5ØØ.

La boucle de jeu est constituée des lignes 5Ø à 25Ø.

5Ø On lit le clavier. Si une touche flèche n'a pas été frappée, il n'y a pas à modifier la position ou l'angle de la raquette et on va en 19Ø.

Les lignes 6Ø à 18Ø modifient la position ou l'angle de la raquette.

6Ø On commence par effacer l'ancienne raquette en y plaçant le caractère B.

7Ø Si on a frappé "flèche en haut", il faut incliner la raquette à droite, J1 est l'incrément angulaire que l'on va ajouter en 11Ø à l'angle R.

8Ø Si on a frappé "flèche en bas", il faut incliner la raquette à gauche.

9Ø Si on a frappé "flèche à droite", il faut déplacer à droite la raquette. I1 est l'incrément que l'on va ajouter en 14Ø à XR.

1ØØ Si on a frappé "flèche à gauche", il faut déplacer à gauche la raquette.

11Ø à 13Ø On modifie l'angle R de la raquette en vérifiant que l'on ne sort pas des limites −RM, +RM.

14Ø à 16Ø On modifie la position XR de la raquette en vérifiant que l'on ne sort pas de l'écran.

17Ø On affiche la raquette. INT(R/J2) vaut −2, −1, Ø, 1 ou 2 selon l'angle de la raquette.

Les lignes 19Ø à 25Ø déplacent la balle.

19Ø On calcule les nouvelles coordonnées XA et YA.

2ØØ Si l'on sort de l'écran par la gauche ou la droite, on va rebondir en 35Ø.

21Ø Si l'on sort de l'écran par le haut, on va aussi rebondir en 35Ø.

22Ø Si l'on sort de l'écran par le bas, on va en 32Ø.

23Ø Si l'on a rencontré une brique, on va en 27Ø.

24Ø On efface l'ancienne balle en X, Y on trace la nouvelle en XA, YA.

25Ø Après avoir transféré (XA, YA) dans (X, Y), on recommence.

Les lignes 27Ø à 31Ø sont exécutées lorsque l'on rencontre une brique.

27Ø On calcule les coordonnées X1, Y1 de l'angle supérieur gauche de la brique rencontrée.

28Ø On efface cette brique.

290 On incrémente le nombre NB de briques cassées et on regarde s'il est égal au nombre total NT de briques.

300 On affiche NB.

310 On nettoie la mémoire et on va rebondir sur la brique en 350.

320 On arrive à cette ligne lorsque l'on sort de l'écran par le bas. Si de plus la balle n'est pas dans la raquette, on va en 1500.

Les lignes 350 à 420 servent à calculer les nouvelles vitesses VX et VY après un rebond sur un plan faisant un angle A avec l'horizontale.

Le sous-programme 500-990 réalise toutes les initialisations nécessaires, variables, constantes et caractères.

Les lignes 1000 à 1100 sont exécutées lorsque l'on a détruit toutes les briques, tandis que les lignes 1500 à 1580 servent lorsque la balle est passée à côté de la raquette.

```
10 REM MUR DE BRIQUES****************
20 :
30 GOSUB 500
40 :
50 T$=KEY$:IF T$<D$ OR T$>A$ THEN 190
60 CURSET XR,YR,3:CHAR B,0,0
70 IF T$=A$ THEN J1=J2:GOTO 110
80 IF T$=B$ THEN J1=-J2:GOTO 110
90 IF T$=C$ THEN I1=I2:GOTO 140
100 IF T$=D$ THEN I1=-I2:GOTO 140
110 R=R+J1:IF R<-RM THEN R=-RM
120 IF R>RM THEN R=RM
130 GOTO 170
140 XR=XR+I1:IF XR<MG THEN XR=MG
150 IF XR>MD-7 THEN XR=MD-7
160 CURSET XR,YR,3
170 CHAR 125+INT(R/J2),0,1
180 :
190 XA=X+VX*DT:YA=Y+VY*DT
200 IF XA<MG OR XA>MD THEN A=PI:GOTO350
210 IF YA<MH THEN A=0:GOTO 350
220 IF YA>MB THEN 320
230 IF POINT(XA,YA)=-1ANDYA<M THEN270
```

```
240 CURSET X,Y,0:CURSET XA,YA,1
250 X=XA:Y=YA:GOTO 50
260 :
270 X1=6*INT(XA/6):Y1=6*INT(YA/6)
280 CURSET X1,Y1,3:FILL 6,1,64
290 NB=NB+1:PRINT:IF NB=NT THEN 1000
300 PRINT"Briques cassees: ";NB:EXPLODE
310 A=0:Z=FRE(""):GOTO 350
320 IF XA<XR-1 OR XA>XR+7 THEN 1500
330 A=R
340 :
350 U=VX*COS(A)+VY*SIN(A)
360 VY=VX*SIN(A)-VY*COS(A):VX=U
370 XA=XA+VX*DT:YA=YA+VY*DT:PING
380 IF XA<MG THEN XA=MG
390 IF XA>MD THEN XA=MD
400 IF YA<MH THEN YA=MH
410 IF YA>MB THEN YA=MB
420 GOTO 240
430 :
500 REM INITIALISATIONS**************
510 TEXT:PAPER 2:INK 0
520 HIRES:RESTORE
530 :
540 A$=CHR$(11)' FLECHE EN HAUT
550 B$=CHR$(10)' FLECHE EN BAS
560 C$=CHR$(9)'  FLECHE A DROITE
570 D$=CHR$(8)'  FLECHE A GAUCHE
580 :
590 PRINT:INPUT"Force (1-5) ";DT
600 IF DT<1 OR DT>5 THEN PING:GOTO590
610 :
620 X=120:Y=195' POSITION BALLE
630 A=(RND(1)-3/2)*PI/2
640 VX=COS(A):VY=SIN(A)'VITESSE BALLE
650 XR=120:YR=192'POSITION RAQUETTE
660 :
670 MG=0:MD=239' LIMITES DE
680 MB=199:MH=0' L'ECRAN
690 M=100' MILIEU DE L'ECRAN
700 :
710 R=0' ANGLE RAQUETTE
```

```
720 J2=PI/36'INCREMENT ANGLE RAQUETTE
730 I2=3'INCREMENT POSITION RAQUETTE
740 RM=J2*2'ANGLE MAXIMAL RAQUETTE
750 :
760 NT=40*10'NOMBRE TOTAL DE BRIQUES
770 NB=0'NOMBRE DE BRIQUES CASSEES
780 :
790 REM DEFINITION RAQUETTES---------
800 DATA 0,0,0,3,12,48,0,0
810 DATA 0,0,0,0,7,56,0,0
820 DATA 0,0,0,0,0,63,0,0
830 DATA 0,0,0,0,56,7,0,0
840 DATA 0,0,0,48,12,3,0,0
850 S=£9800
860 FOR A=S+123*8 TO S+127*8+7
870 READ U:POKE A,U
880 NEXT A
890 :
900 REM DEFINITION CARACTERE BLANC---
910 B=64
920 FOR A=S+64*8 TO S+64*8+7
930 POKE A,63
940 NEXT A
950 :
960 CURSET 0,6,3:FILL 60,40,63
970 CURSET XR,YR,3:CHAR 125,0,1
980 RETURN
990 :
1000 REM GAGNE************************
1010 TEXT:CLS:PRINT:PRINT
1020 PRINT CHR$(4)
1030 PRINT SPC(12);CHR$(27);"N";
1040 PRINT CHR$(27);"AB R A V O"
1050 PRINT:PRINT:PRINT
1060 PRINT SPC(10);CHR$(27);"J";
1070 PRINT CHR$(27);"EPLUS DE BRIQUE"
1080 PRINT CHR$(4):PRINT:PRINT
1090 GOTO 1530
1100 :
```

```
1500 REM PERDU*************************
1510 PRINT"SCORE FINAL:";CHR$(27);"A";
1520 PRINT NB;CHR$(27);"à";"BRIQUES"
1530 PRINT"Voulez-vous rejouer (O/N) "
1540 INPUT T$
1550 IF T$="O" THEN 10
1560 IF T$<>"N" THEN PING:GOTO 1530
1570 TEXT:CALL £F8D0:END
1580 :
```

Note: le caractère "£" est à remplacer par "#".

1.12 BATAILLE NAVALE

Ce jeu vous permet de jouer à la bataille navale en solitaire, l'ORIC se charge de placer les bateaux. Pour tirer, il vous suffit de frapper les coordonnées de l'endroit visé. D'abord, taper une lettre comprise entre A et J, puis un chiffre compris entre Ø et 9.

L'ORIC a placé dans la grille un porte-avion (4 cases), deux croiseurs (trois cases), trois torpilleurs (deux cases) et quatre sous-marins (une case).

Les bateaux ne peuvent se toucher. Après chaque tir, l'ORIC vous indique "à l'eau", "touché" ou "coulé".

Voici quelques explications sur le fonctionnement de ce programme. Tout d'abord, quelques initialisations, de 1Ø à 6Ø.

4Ø CLEAR sert lorsque l'on rejoue, il efface toutes les variables, ce qui est un moyen de les remettre à zéro. Le tableau JEU% sert à y placer les bateaux. Bien que la grille fasse 1Ø×1Ø, ce tableau fait 12×12. Ceci pour simplifier la programmation du jeu. Le tableau COULE% sera progressivement rempli de "Ø" au fur et à mesure que des bateaux seront coulés.

De 8Ø à 15Ø, on dessine la grille de jeu.

1ØØ On place en (1Ø,I) un caractère 16 qui donne à l'écran la couleur noire, jusqu'à ce que l'on rencontre le caractère 2Ø placé par l'instruction suivante, qui redonne à l'écran la couleur bleue.

130 Lorsque I varie de 2 à 11, I+46 varie de 48 à 57, on obtient donc les codes ASCII des caractères "Ø" à "9". I+63 donne lui les codes ASCII des caractères A à J.

Le bloc 16Ø-38Ø place les dix bateaux dans la grille.

2ØØ La variable H permet de décider si le bateau sera horizontal (H=—1) ou vertical (H=Ø).

21Ø X est l'abscisse de la case supérieure gauche du bateau à placer.

22Ø Y est l'ordonnée de cette case.

23Ø XF est l'abscisse de l'autre extrémité du bateau.

24Ø YF est l'ordonnée de l'autre extrémité du bateau.

25Ø à 29Ø On vérifie que les cases situées autour du bâtiment sont bien vides, que le bâtiment n'en touche pas un autre. Si tel n'est pas le cas, on va en 2ØØ, il faut placer ailleurs le bâtiment.

3ØØ à 34Ø On place le bateau dans la grille. Le chiffre des dizaines de 1Ø*TYPE+N est TYPE et indique le type du bâtiment tandis que le chiffre des unités est N et indique le numéro du bâtiment. On peut ainsi stocker deux données dans une même variable.

35Ø On stocke dans COULE% la longueur du bâtiment.

Les lignes 39Ø à 44Ø lisent le tir.

41Ø On lit un caractère. Si ce n'est pas une lettre comprise entre A et K, on recommence.

42Ø On affiche la lettre, et l'on calcule l'abscisse X du tir, comprise entre 1 et 1Ø.

43Ø On lit un caractère. Si ce n'est pas un chiffre compris entre Ø et 9, on recommence.

44Ø On affiche le chiffre, et l'on calcule l'ordonnée Y du tir, comprise entre 1 et 1Ø.

De 46Ø à 49Ø, on regarde si le tir a touché quelque chose. Si non, on place un point sur l'écran.

Le bloc 51Ø à 55Ø est exécuté lorsqu'un bateau est touché.

51Ø On détermine le type du bateau touché.

52Ø On détermine le numéro du bateau touché.

530 On fait disparaître de la grille la case touchée.

540 On décrémente la longueur du bateau.

550 On fait apparaître sur l'écran un chiffre compris entre 1 et 4, qui indique la taille du bateau touché.

De 560 à 590, on affiche le message approprié à la situation. Si la longueur du bateau est réduite à zéro, ce bâtiment vient d'être coulé.

De 610 à 650, on regarde s'il reste des bateaux à flot.

```
10 REM BATAILLE NAVALE***************
20 :
30 REM INITIALISATIONS---------------
40 CLEAR:DIM JEU%(11,11),COULE%(4,4)
50 PAPER 4:INK 7
60 CLS
70 :
80 REM DESSIN DE LA GRILLE-----------
90 FOR I=3 TO 23
100 PLOT 10,I,16:PLOT 31,I,20
110 NEXT I
120 FOR I=2 TO 11
130 PLOT 8,2*I,I+46:PLOT 7+2*I,1,I+63
140 NEXT I
150 :
160 REM ON PLACE LES BATIMENTS-------
170 PLOT 10,25,"Je place les bateaux"
180 FOR TYPE=4 TO 1 STEP-1:V=TYPE-1
190 FOR N=1 TO 5-TYPE
200 H=-1:IF RND(1)>.5 THEN H=0
210 X=1+INT((10+V*H)*RND(1))
220 Y=1+INT((10+V*(NOT H))*RND(1))
230 XF=X-H*V
240 YF=Y-(NOT H)*V
250 FOR I=X-1 TO XF+1
260 FOR J=Y-1 TO YF+1
270 IF JEU%(I,J)<>0 THEN 200
280 NEXT J
290 NEXT I
```

```
300 FOR I=X TO XF
310 FOR J=Y TO YF
320 JEU%(I,J)=TYPE*10+N
330 NEXT J
340 NEXT I
350 COULE%(TYPE,N)=TYPE
360 NEXT N
370 NEXT TYPE
380 :
390 REM DEBUT DU JEU-----------------
400 PLOT 10,25,"Que jouez-vous:      "
410 GET A$:IF A$<"A"ORA$>"K" THEN 400
420 PLOT 28,25,A$:X=ASC(A$)-64
430 GET A$:IF A$<"0"ORA$>"9" THEN 400
440 PLOT 29,25,A$:Y=ASC(A$)-47
450 :
460 IF JEU%(X,Y)<>0 THEN 510
470 PLOT 2*X+9,2*Y+2,"."
480 PLOT 10,25,"   A l'eau.          "
490 WAIT 200:GOTO 400
500 :
510 TYPE=INT(JEU%(X,Y)/10)
520 N=JEU%(X,Y)-10*TYPE
530 JEU%(X,Y)=0
540 COULE%(TYPE,N)=COULE%(TYPE,N)-1
550 PLOT 2*X+9,2*Y+2,TYPE+48
555 :
560 EXPLODE:IFCOULE%(TYPE,N)=0THEN590
570 PLOT 10,25,"    Touche.          "
580 GOTO 610
590 PLOT 10,25,"    Coule !!         "
600 :
610 WAIT 200
620 FOR I=1 TO 4:FOR J=1 TO 4
630 IF COULE%(I,J)<>0 THEN 400
640 NEXT J:NEXT I
650 PLOT 10,25,"   GAGNE !!!         "
```

1.13 MASTER MIND

Le master mind est un jeu de réflexion auquel vous allez pouvoir jouer avec votre ORIC. La règle du jeu est la suivante : L'ORIC choisit une combinaison de pions qu'il vous faut découvrir.

Chacun de ces pions peut être rouge, vert, jaune, bleu, mauve ou cyan. Il vous faut découvrir la couleur et la position de chacun des pions. A chaque coup, vous proposez une combinaison. L'ORIC vous répond en plaçant sur l'écran une petite marque noire pour chaque pion de la bonne couleur à la bonne place. Puis l'ORIC ajoute une petite marque blanche pour chaque pion de la bonne couleur qui n'est pas à la bonne place. A l'aide de ces indications, il faut trouver la combinaison cachée en le moins de coups possible.

Au début du jeu, on vous demande le nombre de pions à découvrir. Tapez 4, 5 ou 6 puis frappez RETURN.

A chaque coup, les touches "flèche à gauche" et "flèche à droite" vous permettent de déplacer le curseur, qui a quatre positions possibles, celles des pions. Les touches R, V, J, B, M et C font apparaître un pion à l'emplacement du curseur. Lorsque vous avez placé dans la grille vos pions, frappez RETURN. Si vous abandonnez, frappez ESC.

Tout autre touche efface le pion sur lequel se trouve le curseur.

Voici quelques précisions sur la structure du programme. Des lignes 3Ø à 4ØØ, on trouve les initialisations nécessaires au jeu.

De 4Ø à 9Ø, on définit les deux caractères représentant le pion et la marque.

De 13Ø à 14Ø, on définit le grand rectangle noir qui servira de grille de jeu.

De 16Ø à 19Ø, on trace le petit rectangle sur lequel le mot "master mind" est inscrit.

De 21Ø à 23Ø, on définit les différentes couleurs.

De 25Ø à 31Ø, on trace un rectangle noir dans lequel on affiche les couleurs possibles.

33Ø L est le numéro de la ligne où se trouve le curseur.

35Ø NC est le nombre de coups joués.

De 37Ø à 4ØØ, on remplit le tableau S avec X nombres compris entre 1 et 6, qui représentent la combinaison à trouver. Ces nombres sont les couleurs des pions.

Le jeu lui-même est constitué des lignes 5ØØ à 1Ø1Ø.

52Ø On déplace de 15 positions vers la droite le curseur pour le placer sur la première colonne de la grille de jeu.

Les lignes 57Ø à 61Ø lisent une touche T$ et effectuent le branchement adéquat.

57Ø Si on a frappé ESC, on abandonne, on va en 117Ø.

58Ø Si on a frappé RETURN, il faut analyser la combinaison proposée, ce que l'on va faire en 7ØØ.

59Ø Si l'on a frappé "flèche à gauche" ou "flèche à droite", on va en 63Ø pour déplacer le curseur.

6ØØ Si l'on a frappé R, V, J, B, M ou C, on va placer un pion en 67Ø.

61Ø Sinon, on place deux blancs pour effacer le pion et son code de couleur.

De 63Ø à 65Ø, on déplace le curseur, à condition de ne pas sortir de la grille. On affiche deux fois T$ en 65Ø : ainsi le curseur se déplace de deux crans.

De 67Ø à 68Ø, on affiche un pion à l'endroit désigné par le curseur. La couleur du pion est I.

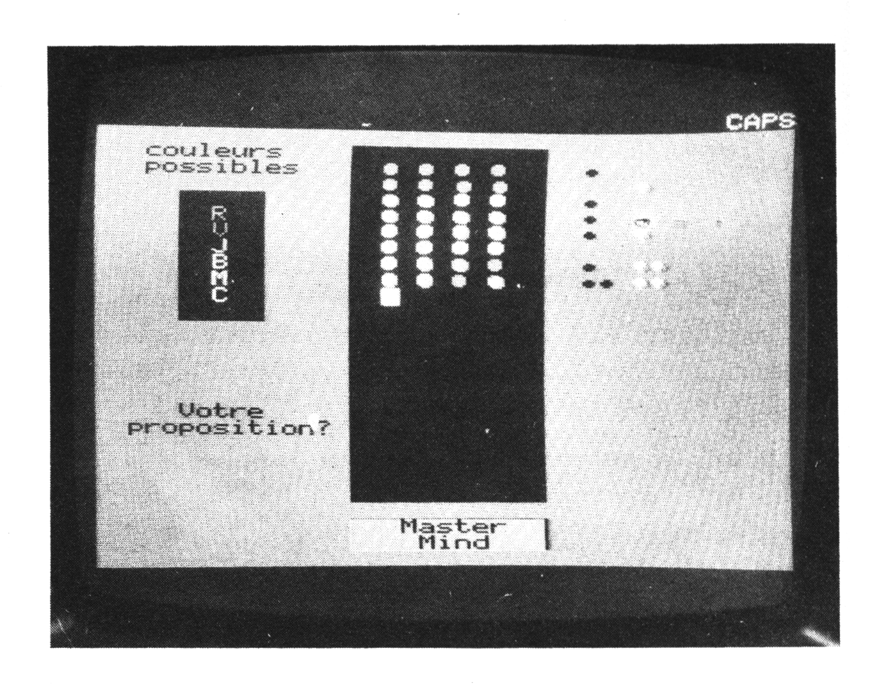
De 7ØØ à 9ØØ, on analyse la proposition. On compte d'abord le nombre N de marques noires, puis ensuite le nombre B de marques blanches.

De 92Ø à 98Ø, on affiche la réponse de l'ORIC. Si on a X marques noires, la combinaison a été découverte et l'on va en 11ØØ.

Si vous n'avez pas trouvé la combinaison et que la grille est pleine, ce n'est pas très brillant et on exécute le bloc 1Ø3Ø-1Ø9Ø.

Si vous avez trouvé, on va en 111Ø, on affiche en double hauteur le mot "BRAVO".

Puis, si vous avez trouvé, ou si vous avez frappé ESC, les lignes 117Ø à 122Ø découvrent la combinaison cachée.




```
10 REM MASTER MIND********************
20 :
30 REM INITIALISATIONS---------------
40 P=125:R=P+1
50 FOR I=£B400+8*P TO £B400+8*R+7
60 READ V:POKE I,V
70 NEXT I
80 DATA 0,28,62,62,62,62,28,0
90 DATA 0,0,12,30,30,12,0,0
100 :
110 IN=0:PA=2:INKIN:PAPERPA:CLS:PRINT
120 INPUT"Nombre de pions";X:IFX<4ORX>6THEN120ELSECL
130 FORI=1TO22:PLOT18+2*X,I,16+PA:NEXT
140 FORI=1TO22:PLOT15,I,16:NEXT
150 :
160 FORI=24TO25:PLOT18+2*X,I,16+PA:NEXT
170 FORI=24TO25:PLOT15,I,17+IN:NEXT
180 PLOT 14+X,24,"Master"
190 PLOT 15+X,25,"Mind"
200 :
```

```
210 DIM C$(6):C$(1)="R":C$(2)="V"
220 C$(3)="J":C$(4)="B":C$(5)="M"
230 C$(6)="C"
240 :
250 PLOT 3,1,"couleurs"
260 PLOT 3,2,"possibles"
270 FOR I=4 TO 11:PLOT10,I,16+PA:NEXT
280 FOR I=4 TO 11:PLOT 5,I,16:NEXT
290 FOR I=5 TO 10
300 PLOT 6,I,I-4:PLOT 7,I,C$(I-4)
310 NEXT
320 :
330 PRINT:PRINT:L=2
340 :
350 NC=1
360 :
370 DIM S(X),F(X),FS(X)
380 FOR I=1 TO X
390 S(I)=INT(6*RND(1))+1
400 NEXT I
500 REM DEBUT DU PROGRAME------------
510 FOR COUP=1 TO 20
520 FOR I=1 TO 15:PRINT CHR$(9);:NEXT
530 PLOT 5,17,"Votre"
540 PLOT 2,18,"proposition?"
550 Z=FRE(""):PLOT 6,14,STR$(COUP)
560 :
570 GET T$:IF T$=CHR$(27) THEN 1170
580 IF T$=CHR$(13) THEN 700
590 IF T$=CHR$(8)ORT$=CHR$(9) THEN630
600 FOR I=1 TO 6:IF T$=C$(I) THEN 670
610 NEXT:PLOTPOS(0)-1,L,"  ":GOTO570
620 :
630 IFPOS(0)<18ANDT$=CHR$(8) THEN 570
640 IFPOS(0)>14+2*XANDT$=CHR$(9)THEN570
650 PRINT T$;T$;:GOTO 570
660 :
670 PLOTPOS(0),L,P:PLOTPOS(0)-1,L,I
680 GOTO 570
690 :
700 PLOT 5,17,"      "
```

```
710 PLOT 2,18,"Voyons...    ":WAIT 120
720 FOR I=1 TO X:F(I)=0:FS(I)=0:NEXT
730 :
740 REM COMBIEN DE NOIRS ?
750 N=0
760 FOR I=1 TO X
770 IF S(I)<>SCRN(14+2*I,L) THEN 790
780 F(I)=1:FS(I)=1:N=N+1
790 NEXT I
800 :
810 REM COMBIEN DE BLANCS ?
820 B=0
830 FOR I=1 TO X
840 IF F(I)=1 THEN 900
850 FOR J=1 TO X
860 IF FS(J)=1 THEN 890
870 IF S(J)<>SCRN(14+2*I,L) THEN 890
880 F(I)=1:FS(J)=1:B=B+1:GOTO 900
890 NEXT J
900 NEXT I
910 :
920 PP=19+2*X:IF N=0 THEN 950
930 PLOT PP,L,0:PP=PP+1
940 FORI=1TON:PLOTPP,L,R:PP=PP+1:NEXT
950 IF B=0 THEN 980
960 PLOT PP,L,7:PP=PP+1
970 FORI=1TOB:PLOTPP,L,R:PP=PP+1:NEXT
980 IF N=X THEN 1100
990 :
1000 PRINT:L=L+1
1010 NEXT COUP
1020 :
1030 CLS:PAPER 1:INK 7
1040 PRINT:PRINT:PRINT:PRINT:PRINT
1050 PRINT SPC(13);"VOUS ETES"
1060 PRINT:PRINT:PRINT CHR$(4)
1070 PRINT SPC(15);
1080 PRINT CHR$(27);"NNUL";CHR$(4)
1090 END
1100 :
```

```
1110 REM TROUVE-------------------
1120 PLOT 4,17,CHR$(10)+"BRAVO !"
1130 PLOT 2,18,CHR$(10)+"  BRAVO !"
1140 PLOT 12,17,8:PLOT 12,18,8
1150 PING
1160 :
1170 PLOT 15,24,16:PLOT18+2*X,24,16+PA
1180 PLOT 15,25,"                "
1190 FOR I=1 TO X
1200 PLOT 14+2*I,24,S(I)
1210 PLOT 15+2*I,24,P
1220 NEXT I
1230 :
1240 PRINT à0,23;
```

Note: le caractère "£" est à remplacer par "#", "à" est à remplacer par " @ ".

1.14 BEATLES

Ce programme n'est pas un jeu, il a simplement pour but celui de vous distraire avec quelques notes de musique, empruntées aux Beatles. Il peut aussi vous montrer comment programmer vos propres morceaux.

Chaque mesure correspond à quatre lignes de DATA. Les doubles zéros que l'on a placé dans les DATA ne sont pas joués, mais sont des points de repère pour pouvoir jouer les reprises.

Les ordres RESTORE permettent de se replacer au début de la partition, tandis que les lectures "bidon" des lignes 2040, 2060 et 2090 permettent respectivement de sauter 1, 1 et 2 mesures.

En modifiant les ordres PLAY et WAIT de la ligne 2540, vous pourrez obtenir d'autres sonorités et tempo.

```
2000 REM MUSIQUE**********************
2010 DU=5000:GOSUB 2510
2020 GOSUB 2510
2030 RESTORE:GOSUB 2510
2040 FOR I=1 TO 25:READ K,N:NEXT
2050 GOSUB 2510
2060 FOR I=1 TO 24:READ K,N:NEXT
2070 GOSUB 2510
```

```
2080 RESTORE:GOSUB 2510
2090 FOR I=1 TO 49:READ K,N:NEXT
2100 GOSUB 2510
2110 WAIT 200:PLAY 0,0,0,0:END
2120 :
2500 REM--------------------------------
2510 FOR I=1 TO 3:READ K,N
2520 IF N=0 THEN RETURN
2530 MUSIC I,K,N,0:NEXT
2540 PLAY 7,0,1,DU:WAIT 15:GOTO 2510
2550 :
2560 :
3000 REM With A Little Help From
3001 REM        My Friends
3002 :
3003 DATA 3,5,3,1,2,8,3,5,3,1,2,8
3004 DATA 3,6,3,1,2,8,3,8,3,1,2,8
3005 DATA 3,8,2,12,2,8,3,8,2,12,2,8
3006 DATA 3,6,2,12,2,8,3,5,2,12,2,8
3007 DATA 3,3,2,10,2,6,3,3,2,10,2,6
3008 DATA 3,5,2,10,2,6,3,6,2,10,2,6
3009 DATA 3,6,2,10,2,3,3,6,2,10,2,3
3010 DATA 3,6,2,10,2,3,3,6,2,10,2,3
3011 DATA 3,5,2,10,2,6,3,5,2,10,2,6
3012 DATA 3,3,2,10,2,6,3,3,2,10,2,6
3013 DATA 3,3,2,8,2,6,3,3,2,8,2,6
3014 DATA 3,1,2,8,2,6,3,3,2,8,2,6
3015 DATA 3,5,2,1,1,1,3,5,2,1,1,1
3016 DATA 3,1,2,8,1,1,3,1,2,8,1,1
3017 DATA 3,1,2,10,1,1,3,1,2,10,1,1
3018 DATA 3,1,2,12,1,1,3,1,2,12,1,1
3019 :
3020 DATA 3,5,3,1,2,8,3,5,3,1,2,8
3021 DATA 3,6,3,1,2,8,3,8,3,1,2,8
3022 DATA 3,8,2,12,2,8,3,8,2,12,2,8
3023 DATA 3,6,2,12,2,8,3,5,2,12,2,8
3024 DATA 3,3,2,10,2,6,3,3,2,10,2,6
3025 DATA 3,5,2,10,2,6,3,6,2,10,2,6
3026 DATA 3,6,2,10,2,3,3,6,2,10,2,3
3027 DATA 3,6,2,10,2,3,3,6,2,10,2,3
3028 DATA 3,5,2,10,2,6,3,5,2,10,2,6
```

```
3029 DATA 3,3,2,10,2,6,3,3,2,10,2,6
3030 DATA 3,3,2,8,2,6,3,3,2,8,2,6
3031 DATA 3,1,2,8,2,6,3,3,2,8,2,6
3032 DATA 3,5,2,8,2,1,3,5,2,8,2,1
3033 DATA 3,5,2,8,2,1,3,5,2,8,2,1
3034 DATA 3,8,3,8,3,8,3,1,3,1,2,1
3035 DATA 3,5,3,5,3,5,3,5,3,5,3,5
3036 :
3037 DATA 3,3,2,11,2,6,3,1,2,11,2,6
3038 DATA 3,1,2,11,2,6,3,1,2,11,2,6
3039 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3040 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3041 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3042 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3043 DATA 3,8,3,8,3,8,3,1,3,1,3,1
3044 DATA 3,5,3,5,3,5,3,5,3,5,3,5
3045 DATA 3,3,2,11,2,6,3,1,2,11,2,6
3046 DATA 3,1,2,11,2,6,3,1,2,11,2,6
3047 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3048 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3049 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3050 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3051 DATA 3,8,3,8,3,8,3,1,3,1,3,1
3052 DATA 3,3,3,3,3,3,3,5,3,5,3,5
3053 :
3054 DATA 3,3,2,10,2,6,3,1,2,10,2,6
3055 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3056 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3057 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3058 DATA 0,0
3059 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3060 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3061 DATA 2,10,2,10,2,10,2,10,2,10
3062 DATA 2,10,2,12,2,12,2,12,2,12
3063 DATA 2,12,2,12,0,0
3064 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3065 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3066 DATA 3,8,3,8,3,8,3,8,3,8,3,8
3067 DATA 3,10,3,10,3,10,3,10,3,10
3068 DATA 3,10,0,0
3069 DATA 3,1,2,8,2,5,3,1,2,8,2,5
```

```
3070 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3071 DATA 3,1,2,8,2,5,3,1,2,8,2,5
3072 DATA 0,0,0,0,0,0,0,0,0,0,0,0
3073 DATA 4,1,3,5,3,1,4,1,3,5,3,1
3074 DATA 4,1,3,5,3,1,4,1,3,5,3,1
3075 DATA 4,1,3,5,3,5,4,1,3,5,3,5
3076 DATA 4,1,3,5,3,5,4,1,3,5,3,5
3077 :
3078 DATA 3,12,3,7,2,3,3,12,3,7,2,3
3079 DATA 3,10,3,7,2,3,3,10,3,7,2,3
3080 DATA 3,10,3,7,2,3,3,10,3,7,2,3
3081 DATA 3,3,3,1,2,3,3,3,3,1,2,3
3082 DATA 3,5,3,1,2,8,3,5,3,1,2,8
3083 DATA 3,5,3,1,2,8,3,5,3,1,2,8
3084 DATA 3,3,2,11,2,6,3,5,2,11,2,6
3085 DATA 3,3,2,11,2,6,3,3,2,11,2,6
3086 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3087 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3088 DATA 3,8,3,1,2,6,3,8,3,1,2,6
3089 DATA 3,10,3,1,2,6,3,10,3,1,2,6
3090 DATA 4,1,3,5,3,1,4,1,3,5,3,1
3091 DATA 4,1,3,5,3,1,4,1,3,5,3,1
3092 DATA 4,1,3,5,3,5,4,1,3,5,3,5
3093 DATA 4,1,3,5,3,5,4,1,3,5,3,5
3094 DATA 3,12,3,7,2,3,3,12,3,7,2,3
3095 DATA 3,10,3,7,2,3,3,10,3,7,2,3
3096 DATA 3,10,3,7,2,3,3,10,3,7,2,3
3097 DATA 3,3,3,1,2,3,3,3,3,1,2,3
3098 DATA 3,5,3,1,2,8,3,5,3,1,2,8
3099 DATA 3,5,3,1,2,8,3,5,3,1,2,8
3100 DATA 3,3,2,11,2,6,3,5,2,11,2,6
3101 DATA 3,3,2,11,2,6,3,3,2,11,2,6
3102 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3103 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3104 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3105 DATA 3,1,2,10,2,6,3,1,2,10,2,6
3106 DATA 0,0
```

1.15 PENDU

Le but de ce jeu est de deviner un mot, que l'ORIC tire au sort. A chaque coup, vous proposez une lettre. Si cette lettre n'est pas contenue dans le mot, un membre du pendu est ajouté. Vous avez le droit à six erreurs.

Tout d'abord, le programme vous demande le nom de la rubrique d'où il va tirer le mot à chercher. Les trois premières rubriques, MUSICIENS, SPORTIFS et HISTOIRE contiennent des noms d'hommes célèbres. La rubrique BOISSONS contient des noms de boissons, généralement alcolisées. La rubrique ACTEURS contient aussi des noms de personnages célèbres. Enfin, STOP permet de sortir du programme.

Si vous ajoutez des noms à une rubrique, n'oubliez pas de modifier en conséquence la ligne 3Ø ou 35. En effet, S(Ø) contient le nombre de noms de la première rubrique, S(1) contient le nombre de noms de la seconde rubrique, etc...

Voici quelques explications sur ce programme. Tout d'abord, de la ligne 5 à la ligne 35, quelques initialisations:

15 L$ sera utilisé pour stocker, dans l'ordre alphabétique, les lettres déjà proposées par le joueur. P$ contiendra les caractères qui figurent les membres du pendu, D$ contient la définition du pendu, MOT$ contient les lettres du mot découvertes et S contient le nombre de mots de chaque rubrique.

2Ø On range dans D$ les caractères qui figureront les membres du pendu.

3Ø à 35 On place dans S le nombre de mots de chaque rubrique.

De 45 à 135, on demande au joueur la rubrique qu'il choisit, on lit sa réponse, et l'on se branche en fonction de sa réponse.

De 14Ø à 17Ø, l'ORIC tire le mot que vous allez chercher.

16Ø à 165 On saute tous les mots correspondants aux sujets situés avant le sujet choisi.

17Ø On lit N mots du sujet choisi. A la sortie de cette boucle, MOT$ contient le dernier mot lu.

De 175 à 22Ø, on effectue quelques initialisations.

18Ø On remplit de blancs le tableau des lettres déjà proposées.

185 On remplit de blancs le tableau des membres du pendu.

19Ø Idem avec le tableau qui représente le mot sur l'écran.

2ØØ à 215 Dans le tableau MOT$, on place un tiret à la place de chaque lettre du mot à chercher. Les caractères qui ne sont pas des lettres (blancs, tirets, etc...) sont placés tels quels.

22Ø On construit une chaîne d'étoiles égale à la longueur du mot à chercher plus 4.

La boucle de jeu est formée par le bloc 225 à 425.

225 à 330 On affiche le jeu.

33Ø Si F=1, le jeu est terminé. On lit une touche quelconque et l'on se branche au début pour recommencer.

335 à 345 On lit un caractère proposé, en vérifiant qu'il s'agit bien d'une lettre.

De 35Ø à 425, c'est l'ORIC qui travaille.

355 On vérifie que la lettre L$ n'a pas déjà été proposée. Si c'est le cas, on va en 425.

365 à 385 La lettre L$ est insérée dans la liste des lettres déjà proposée, de façon à ce que cette liste soit classée dans l'ordre alphabétique.

39Ø à 4Ø5 Les lettres du mot égales à la lettre proposée sont placées dans le tableau MOT$. En sortie de cette boucle, OK vaut 1 si la lettre proposée était bien dans le mot.

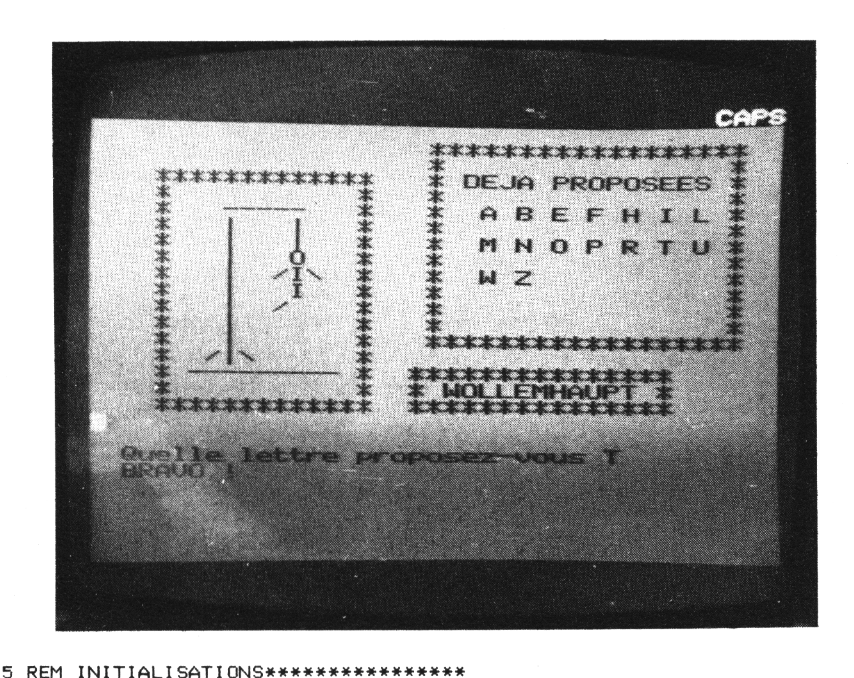
4Ø5 Si le mot a été trouvé, on positionne F à 1.

41Ø Si la lettre proposée était bonne, on efface le message précédent s'il y en avait un.

415 La lettre proposée n'était pas bonne, on ajoute un membre au pendu.

42Ø Là, vous êtes très mal !

Les mots sont stockés dans une liste DATA, en 1ØØØ pour la première rubrique, en 2ØØØ pour la seconde, en 3ØØØ pour la troisième, etc...




```
5 REM INITIALISATIONS****************
10 CLEAR
15 DIM L$(25),P$(6),D$(6),MOT$(20),S(5)
20 D$(0)="O":D$(1)="I":D$(2)="I":D$(3)="/"
25 D$(4)="ç":D$(5)="/":D$(6)="ç"
30 S(0)=25:S(1)=25:S(2)=25:S(3)=25
35 S(4)=25:S(5)=100
40 REM CHOIX DU SUJET----------------
45 TEXT:CLS:PAPER2:INK0:PRINT:L=0:F=0:I=0
50 PRINTSPC(10);"JEU DU PENDU":PRINT
55 PRINT"Les sujets sont -MUSICIENS"
60 PRINT"                -SPORTIFS"
65 PRINT"                -HISTOIRE"
70 PRINT"                -BOISSONS"
75 PRINT"                -CAPITALES"
80 PRINT"                -ACTEURS"
85 PRINT"                -SEXE"
90 PRINT"                -STOP"
95 PRINT:PRINT:INPUT"VOTRE CHOIX ";CH$
100 IF CH$="MUSICIENS" THEN CH=0:GOTO 145
105 IF CH$="SPORTIFS" THEN CH=1:GOTO 145
110 IF CH$="HISTOIRE" THEN CH=2:GOTO 145
115 IF CH$="BOISSONS" THEN CH=3:GOTO 145
120 IF CH$="CAPITALES" THEN CH=4:GOTO 145
125 IF CH$="ACTEURS" THEN CH=5:GOTO 145
127 IF CH$="SEXE" THEN 10000
130 IF CH$="STOP" THEN PRINT"AU REVOIR.":END
135 PRINT"RATE !!!!":GOTO 95
140 REM CHOIX DU MOT A TROUVER-------
145 N=INT(RND(1)*S(CH))+1
150 RESTORE
155 IF CH=0 THEN 170
160 FOR I=1 TO CH:FOR J=1 TO S(I-1)
165 READ MOT$:NEXT J,I
```

```
170 FOR I=1 TO N:READ MOT$:NEXT
175 REM PREPARATION DE L'AFFICHAGE---
180 FOR I=0 TO 25:L$(I)=CHR$(32):NEXT
185 FOR I=0 TO 6:P$(I)=CHR$(32):NEXT
190 FOR I=0 TO 20:MOT$(I)=CHR$(32):NEXT
195 CLS:FIN=LEN(MOT$)
200 FOR I=1 TO FIN:V$=MID$(MOT$,I,1)
205 IF V$=>"A"ANDV$<="Z" THEN MOT$(I-1)="-":GOTO 215
210 MOT$(I-1)=V$:L=L+1
215 NEXT:C$=""
220 FOR I=1 TO LEN(MOT$)+4:C$=C$+"*":NEXT
225 REM AFFICHAGE DU JEU-------------
230 PRINT CHR$(30);
235 PRINT
240 PRINTSPC(18);"******************"
245 PRINTSPC(18);"*                *"
250 PRINT"  *************   * DEJA PROPOSEES *"
255 PRINT"  *           *   *";SPC(16);"*"
260 PRINT"  *   -----   *   *  ";L$(0);" ";L$(1);" ";L$(2);" ";L$(3);" ";
262 PRINT L$(4);" ";L$(5);" ";L$(6);" *"
265 PRINT"  *   ù   ù   *   *";SPC(16);"*"
270 PRINT"  *   ù   ù   *   *  ";L$(7);" ";L$(8);" ";L$(9);" ";L$(10);" ";
272 PRINTL$(11);" ";L$(12);" ";L$(13);" *"
275 PRINT"  *   ù   ";P$(0);"   *   *";SPC(16);"*"
280 PRINT"  *   ù  ";P$(3);P$(1);P$(4);"  *   *  ";L$(14);" ";L$(15);" ";
282 PRINTL$(16);" ";L$(17);" ";L$(18);" ";L$(19);" ";L$(20);" *"
285 PRINT"  *   ù   ";P$(2);"   *   *";SPC(16);"*"
290 PRINT"  *   ù  ";P$(5);" ";P$(6);"  *   *  ";L$(21);" ";L$(22);" ";
292 PRINTL$(23);" ";L$(24);" ";L$(25);"      *"
295 PRINT"  *   ù       *   *";SPC(16);"*"
300 PRINT"  *   ù       *   ******************"
305 PRINT"  *  /ùç      *"
310 PRINT"  * --------- *  ";C$
315 PRINT"  *           *  * ";
320 FOR I=1 TO FIN:PRINT MOT$(I-1);:NEXT
325 PRINT" *"
330 PRINT"  *************  ";C$:IF F=1 THEN GET A$:GOTO 5
335 PRINT:PRINT: PRINTCHR$(14);"Quelle lettre proposez-vous ";
340 GET L$:IFL$<"A"ORL$>"Z"THEN340
345 PRINT L$
350 REM TRAITEMENT------------------
355 FOR I=0 TO 25:IF L$(I)=L$ THEN 425
360 NEXT:OK=0
365 FOR I=0 TO 25:IF L$<L$(I) THEN 380
370 IF L$(I)=" " THEN 380
375 NEXT
380 FOR J=25 TO I+1 STEP-1:L$(J)=L$(J-1):NEXT
385 L$(I)=L$:TR=1
390 FOR I=1 TO FIN:V$=MID$(MOT$,I,1)
395 IF V$=L$ THEN MOT$(I-1)=L$:OK=1:L=L+1:GOTO 405
400 TR=0
405 NEXT:IF L=LEN(MOT$) THEN PRINT"BRAVO !":F=1:GOTO 225
410 IF OK THEN PRINT SPC(21):GOTO 225
415 P$(K)=D$(K):K=K+1:IF K<7 THEN PRINT SPC(21):GOTO 225
420 PRINT"Le mot etait ";MOT$:F=1:GOTO 225
425 PRINT "DEJA PROPOSEE":GOTO 225
1000 DATA BERLIOZ,BACH,BEETHOVEN,BIZET,BRAHMS
1001 DATA CHOPIN,BUXEHUDE,HUMMEL,MOUSSORGSKY,PACHELBEL
1002 DATA JOSQUIN DES PRES,WAGNER,MENDELSSOHM,SCARLATTI,WOLLEMHAUPT
1003 DATA CZERNY,BRAUNGARDT,RAMEAU,ROSSINI,SCHUBERT
1004 DATA SCHUMANN,WEBER,MOZART,VIVALDI,OFFENBACH
2000 DATA TABARLY,PROST,PLATINI,KILLY,BORG
```

```
2001 DATA RUSSEL,"CASSIUS CLAY","MADAME CLAUDE",DARNICHE,"NIKI LAUDA"
2002 DATA CAW,PIRONI,LAFFITE,ZITRONE,CHICHESTER
2003 DATA JAZY,MIMOUN,ANQUETIL,BOBET,HINAULT
2004 DATA MERCKX,MOSER,POULIDOR,COPI,RAAS
3000 DATA "DE GAULLE",CHARLEMAGNE",VERCINGETORIX",ATTILA,NAPOLEON
3001 DATA HITLER,WASHINGTON,CHURCHILL,PASTEUR,FLEMMING
3002 DATA PETAIN,"HIRO HITO","PEPIN LE BREF",CLOVIS
3003 DATABERTHE AU GRAND PIED,"BLANCHE DE CASTILLE",ROOSVELT,KENNEDY,DESCHANEL
3004 DATA"LA POMPADOUR","MUSTAPHA KEMAL",POMPIDOU,REAGAN,TCHERNENKO
4000 DATA CONTREXEVILLE,ARMAGNAC,WHISKY,VODKA,GIN
4001 DATA FRAMBOISE,GENEPI,CHARTREUSE,SAUTERNES,JULIENAS
4002 DATA"CHATEAU PETRUS",CHAMPAGNE,COGNAC,RIESLING,GEWURSTRAMINER
4003 DATA POIRE,GNOLE,SCHNAPS,PEINTURE,TRICHLORETYLENE
4004 DATA BRANDY,MADERE,PREFONTAINES,CIDRE,SAKE
5000 DATA PARIS,LONDRES,BONN,ROME,MADRID
5001 DATA WASHINGTON,TRIPOLI,RYAD,"LE CAIRE",RABAT
5002 DATA TANANARIVE,MOSCOU,ANKARA,PEKIN,BANGKOK
5003 DATA TEHERAN,BAGDAD,MONTREAL,RANGOON,TOKIO
5004 DATA BRASILIA,MEXICO,GEORGETOWN,CANBERRA,BAMAKO
6000 DATA HEPBURN,BOGART,BRANDO,GIRARDOT,MASTROIANNI
6001 DATA CASSEL,DELON,SINATRA,CECCALDI,GALABRU
6002 DATA MEURISSE,FERREOL,NOIRET,MONTANT,BURTON
6003 DATA DARC,WELLS,DENEUVE,BELMONDO,WINDMARK
6004 DATA PIAT,DARRAS,NEWMAN,SELLERS,VANNECK
6005 DATA LOREN,MOREAU,FLYNN,DEPARDIEU,BRYNNER
6006 DATA LOLLOBRIDGIDA,CONSTANTIN,ANDRESS,MARIELLE,KEATON
6007 DATA CHAPLIN,LAUREL,HARDY,JOBERT,BLANCHE
6008 DATA DIFILHO,BLIER,MONROE,POIRET,BOURVIL
6009 DATA FERNANDEL,LEWIS,YANNE,DAUPHIN,DEREK
6010 DATA DUTRONC,MANFREDI,WAYNE,USTINOV,CURTIS
6011 DATA SERRAULT,BALASKO,DEWAERE,BALUTIN,TOGNAZZI
6012 DATA PICCOLI,PACOME,HUPPERT,ASTAIRE,PRESLEY
6013 DATA ADAMO,HOLDEN,GABIN,BRIALY,SIGNORET
6014 DATA ADJANI,CARREL,VENTURA,HANIN,SCHNEIDER
6015 DATA LAFONT,GLASER,TRINTIGNANT,TAYLOR,GASSMAN
6016 DATA FONDA,SHARIF,SEBERG,ROCHEFORT,LANVIN
6017 MITCHUM,HESTON,FOSSEY,AUCLAIR,QUINN
6018 DATA JOUVET,BRASSEUR,MARAIS,MORGAN,HOFFMAN
6019 DATA CARMET,BARDOT,NEVILLE,KELLER,CHAKIRIS
10000 REM COMMENTAIRE POUR SEXE------
10005 CLS:PRINT:PRINT
10010 PRINT"JE NE JOUE PAS AVEC VOUS"
10015 PRINT"ESPECE DE SATYRE !!!!"
10020 PAPER 6:EXPLODE:WAIT 200
10025 PRINT:PRINT
10030 PRINT"JE REFUSE DE JOUER AVEC"
10035 PRINT"UN TEL OBSEDE."
10040 PAPER 1
10045 FOR I=1 TO 10:SHOOT:WAIT10:NEXT
10050 PRINT:PRINT:PRINT
10055 PAPER 1
10060 PRINT"POUR QUI ME PRENEZ-VOUS"
10065 PRINT"DEBILE !!!!!!!!!!!!!"
10070 FOR I=1 TO 7
10075 ZAP:WAIT 10:EXPLODE:WAIT 15
10080 NEXT
10085 FOR I=1TO 20
10090 SHOOT:WAIT 8
10095 NEXT:PRINTCHR$(6)
10100 PLAY 7,7,4,300
```

Note: le caractère "ù" est à remplacer par "|", tandis que "ç" est à remplacer par "\".

2

Quelques renseignements utiles pour programmer vos jeux

2.1 GESTION DE LA MÉMOIRE DE L'ORIC/ATMOS

Lorsque l'on programme en basic, on n'a pas à se préoccuper de la façon dont la mémoire est gérée, utilisée. Par contre, dès que l'on introduit quelques instructions de langage machine, ou même plus simplement, dès que l'on utilise PEEK, DEEK, POKE ou DOKE, il faut malheureusement entrer dans ces détails sordides que sont une gestion mémoire.

Tout d'abord, voici la carte mémoire de l'ORIC :

en mode HIRES

	FFFF-65535
ROM : Basic de l'ORIC	
	CØØØ-49152
rien	
	BFEØ-4912Ø
écran	
	AØØØ-4Ø96Ø
caractères semi-graphiques	
	9CØØ-39936
caractères standards	
	98ØØ-38912
programme Basic (début en #5Ø1)	
	5ØØ-128Ø
de 4ØØ à 42Ø : rien	
	4ØØ-1Ø24
adresses d'entrées/sorties	
	3ØØ-768
utilisé par le Basic	
	2ØØ-512
pile du 65Ø2	
	1ØØ-256
page Ø utilisée	
	Ø-Ø

en mode TEXT

	FFFF-65535
ROM : Basic de l'ORIC	
	CØØØ-49152
rien	
	BFEØ-4912Ø
écran	
	BB8Ø-48ØØØ
caractères semi-graphiques	
	B8ØØ-471Ø4
caractères standards	
	B4ØØ-46Ø8Ø
rien	
	9FØØ-4Ø7Ø4
programme Basic (début en #5Ø1)	
	5ØØ-128Ø
de 4ØØ à 42Ø : rien	
	4ØØ-1Ø24
adresses d'entrées/sorties	
	3ØØ-768
utilisé par le Basic	
	2ØØ-512
pile du 65Ø2	
	1ØØ-256
page Ø utilisée	
	Ø-Ø

Les adresses sont données en hexadécimal puis en décimal.

a) La page zéro. #Ø à #FF

Par construction du microprocesseur 65Ø2, qui équipe votre ORIC, les octets d'adresse Ø à 255 peuvent être lus ou écrits plus rapidement que les autres. Dans un souci d'efficacité, il est conseillé de placer dans cette zone les quantités auxquelles le microprocesseur accède le plus souvent. Le Basic utilise la plupart de ces octets. Si vous écrivez un sous-programme en langage machine utilisé depuis un programme Basic, il est conseillé d'employer les octets inutilisés par le Basic. C'est le cas de la zone d'adresse Ø à 11 (#Ø à #B).

b) La pile du microprocesseur. #1ØØ à #1FF

A chaque fois qu'une instruction JSR, PHA ou PHP est exécutée par le 65Ø2, l'adresse de retour ou l'accumulateur ou le registre d'état est empilé dans cette zone. Il est très fortement déconseillé d'écrire dans cette zone, sauf si vous aimez vivre dangereusement.

c) Zone de travail du Basic. #2ØØ à #2FF

Le Basic a besoin de mémoire pour travailler. Il utilise la page zéro, la zone #2ØØ à #2FF et la zone #421 à 4FF.

d) Les adresses d'entrées/sorties. #3ØØ à #3FF

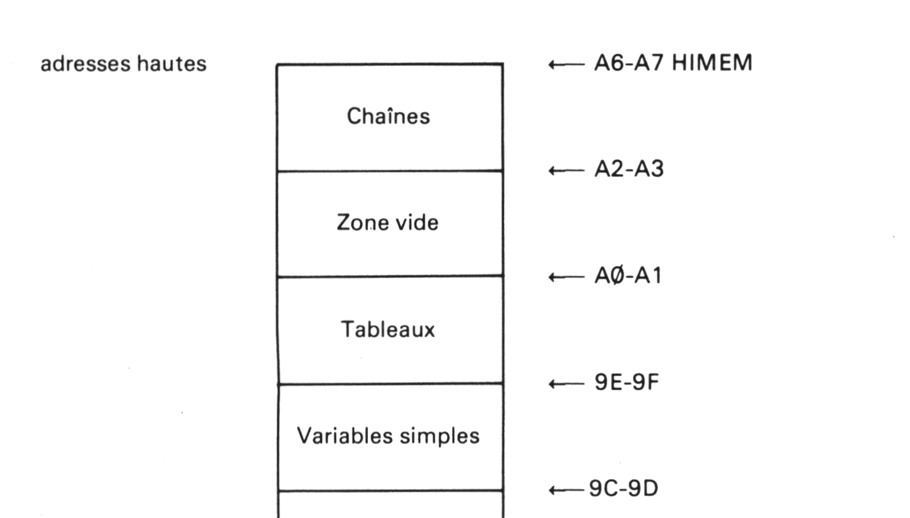
Il n'y a pas de mémoire à ces adresses, mais tous les dispositifs externes à l'ensemble microprocesseur-mémoire. On y trouve entre autres le clavier, le magnétophone, le synthétiseur, etc...

e) Zone de travail du Basic. #421 à #4FF

Il est à noter que la zone #4ØØ-#42Ø est inutilisée. Vous pouvez l'employer pour y placer des sous-programmes en langage machine.

f) La zone programme Basic. #5ØØ à #97FF

On y trouve d'une part le programme Basic lui-même, d'autre part les variables. Plus précisément, on a dans cette zone, l'organisation suivante :



— En partant des adresses basses, on trouve d'abord le programme Basic lui-même. L'adresse-mémoire du début du programme est rangée en 9A-9B, elle vaut normalement 5Ø1. En tapant PRINT HEX$(DEEK(#9A)) vous obtenez cette valeur.

— Ensuite on trouve mélangées les variables entières, les variables réelles et les noms des variables chaînes. L'adresse-mémoire du début de cette zone est rangée en 9C-9D, on peut la lire en exécutant PRINT HEX$(DEEK(#9C)).

Comme les zones programme et variables simples sont contiguës, on peut considérer que cette adresse est aussi celle de la fin du programme.

Chaque variable simple occupe sept octets de mémoire. Les variables sont placées en mémoire au fur et à mesure qu'elles sont rencontrées, tous types mélangés.

* *Variable réelle* : nombre flottant à 5 octets

		EXP		M	
P1	P2	exp+128		mantisse	

P1 et P2 sont les deux premiers caractères du nom de la variable, en code ASCII positif ($P1<128$ et $P2<128$). Si le nom n'a qu'une lettre, P2 vaut Ø.

Le troisième octet contient l'exposant plus 128, les quatre derniers octets constituent la mantisse, qui est comprise entre 1 et 1/2. Le nombre stocké vaut $M.2^{(EXP-128)}$.

* *Variable entière*: nombre à deux octets en complément à 2

N1	N2	poids fort	poids faible	Ø	Ø	Ø

N1 et N2 sont les deux premiers caractères du nom de la variable, en code ASCII négatif ($N1>128$ et $N2>128$).

* *Variable chaîne*: chaîne de Ø à 255 caractères

P1	N2	nb. carac.	ad. bas	ad. haut	Ø	Ø

Le premier caractère du nom de la variable est stocké en code ASCII positif, tandis que le second est stocké en code ASCII négatif ($P1<128$ et $N2\geqslant128$). Le troisième octet contient la longueur de la chaîne, Ø si elle est vide. Puis l'on trouve, sur deux octets, l'adresse-mémoire de la chaîne. On a d'abord l'octet de poids faible, puis l'octet de poids fort de cette adresse.

La chaîne elle-même peut se trouver soit dans la zone de stockage des chaînes, en haut de la mémoire, soit tout simplement à l'intérieur du programme Basic.

Exemples:

Si l'on a:

```
1ØØ A$=B$+C$
```

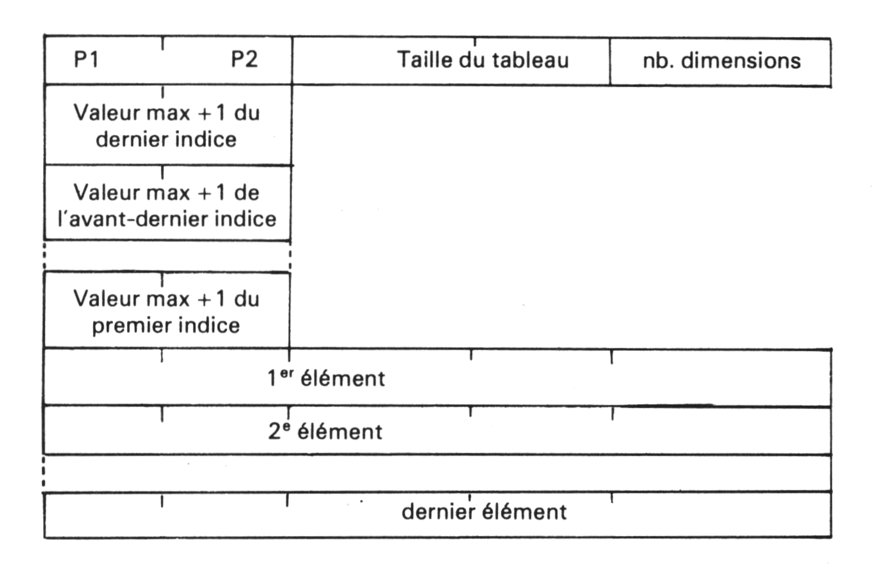
Lorsque cette ligne sera exécutée, la chaîne correspondant à A$ sera stockée en haut de la mémoire. Par contre, si l'on a :

```
200 A$="123456789"
```

la chaîne correspondant à A$ est la suite de caractères 123456789, elle existe à l'intérieur du programme, il est inutile d'aller la recopier en haut de la mémoire.

— Après les variables simples, on trouve les tableaux. L'adresse de début de cette zone est stockée en 9E-9F. Pour la lire, exécutez PRINT HEX$(DEEK(#9E)). On peut aussi considérer que cette adresse est celle de la fin des variables simples, car les zones correspondant aux variables simples et aux tableaux sont contiguës.

* *Tableau réel*



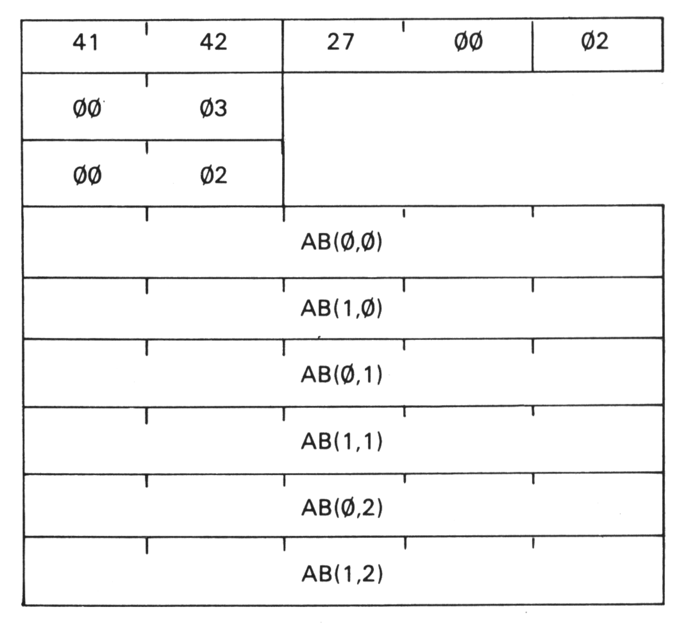
On trouve d'abord les deux premiers caractères du nom du tableau P1 et P2, en code ASCII positif. Ensuite on trouve sur deux octets, la taille du tableau, c'est-à-dire le nombre total d'octets utilisés pour stocker le tableau. On trouve d'abord l'octet de poids faible, puis l'octet de poids fort de ce nombre. Sur un octet, on trouve ensuite le nombre de dimensions, c'est-à-dire le nombre d'indices du tableau. On trouve

ensuite pour chaque indice, en partant du dernier jusqu'au premier, la valeur maximale plus un de l'indice. Chacun de ces nombres est codé sur deux octets, l'octet de poids fort étant le premier. Les éléments du tableau, les réels eux-mêmes sont ensuite rangés en faisant d'abord varier l'indice le plus à gauche.

Exemple :

DIM AB(1,2)

définit un tableau à six éléments et deux dimensions, qui est stocké sous la forme :



— 41 et 42 sont les codes ASCII des lettres A et B (65 et 66 en base 1Ø).

— ØØ27 est la taille du tableau (39 en base 1Ø).

— ØØØ3 est la valeur maximale plus un du deuxième indice.

— ØØØ2 est la valeur maximale plus un du premier indice.

* *Tableau entier*

La structure est analogue à celle d'un tableau réel, sauf qu'un élément occupe deux octets seulement contre 5 pour un réel.

N1 N2	Taille du tableau	nb. dimensions
Valeur max +1 du dernier indice		
Valeur max +1 de l'avant dernier indice		
⋮		
Valeur max +1 du premier indice		
1er élément		
⋮		
dernier élément		

Les deux premiers caractères du nom du tableau N1 et N2 sont en code ASCII négatif.

Pour chaque élément, on stocke l'octet de poids fort, puis celui de poids faible.

* *Tableau chaîne*

La structure est aussi analogue à celle d'un tableau réel sauf qu'un élément occupe trois octets seulement.

P1 N2	Taille du tableau	nb. dimensions
Valeur max +1 du dernier indice		
Valeur max +1 de l'avant dernier indice		
⋮		
Valeur max +1 du premier indice		
1er élément		
⋮		
dernier élément		

Le premier caractère du nom du tableau, P1, est stocké en code ASCII positif, tandis que le second est en code ASCII négatif.

Pour chaque chaîne on stocke sa longueur, l'octet de poids faible puis l'octet de poids fort de l'adresse de la chaîne.

L'adresse de la fin de la zone des tableaux est stockée en AØ-A1. En exécutant PRINT HEX$(DEEK(#AØ)), on peut lire cette adresse en hexadécimal.

— Les chaînes sont-elles stockées en partant de HIMEM, qui est en A6-A7, puis en descendant. A2-A3 contient l'adresse de l'autre extrémité de cette zone.

— Entre la zone des tableaux et celle des chaînes, on a une zone vide, inoccupée. Au fur et à mesure qu'un programme s'exécute, des variables, des tableaux et des chaînes sont créés, et cette zone vide rétrécit par les deux bouts.

Si donc vous désirez utiliser de la mémoire, soit pour y placer des sous-programmes assembleurs ou des données structurées par vous-même, sans risquer de détruire le Basic ou d'être détruit par lui, vous avec plusieurs possibilités.

* Utiliser la zone 4ØØ à 42Ø.

* Si cela ne suffit pas on peut, avant que le programme Basic ne crée une chaîne, exécuter une instruction HIMEM qui abaisse le contenu de A6-A7. Les octets situés au-dessus de ce plafond sont alors disponibles.

* On peut aussi, avant que le programme n'ait créé une variable ou un tableau, exécuter un ordre DOKE en 9C pour augmenter l'adresse qui y est contenue, puis utiliser les octets situés entre la fin du programme Basic et cette nouvelle adresse. Ceci a un avantage: comme l'ordre CSAVE recopie sur cassette la zone dont l'adresse de début est en 9A-9B, dont l'adresse de fin est en 9C-9D, un seul ordre CSAVE permet de sauvegarder en même temps le programme Basic, et par exemple, des sous-programmes machine placés à sa suite. Mais attention, la médaille a un revers: Il devient impossible de modifier le programme Basic. En effet, lorsque vous modifiez un programme, vous lui ajoutez ou vous lui retirez des caractères. Aussi, l'ORIC se doit de déplacer dans la mémoire une partie du programme Basic, depuis l'endroit où s'est produite la modification, jusqu'à sa fin. Or, il se trouve que chaque ligne de programme Basic en mémoire contient l'adresse de la ligne suivante. Dans tout le bloc qui a été déplacé, l'ORIC doit recalculer ces adresses. Et en faisant cela, il se plante lorsque 9C-9D ne contient pas véritablement l'adresse de fin du programme Basic.

g) La zone de stockage des caractères

Pour chacun des deux jeux de caractères de l'ORIC, une zone de 1 Koctet est réservée, ces deux zones sont contiguës. Deux choses sont à remarquer :

— Les caractères ne sont pas situés au même endroit en mode TEXT et en mode HIRES, à chaque fois que vous passez d'un mode à l'autre, l'ORIC les déplace. Aussi, il faut prendre garde à ceci lorsque l'on redéfinit un caractère :

La définition du caractère normal de code ASCII "A" débute à l'adresse 46Ø8Ø+8*A en mode TEXT (ou LORES), tandis qu'elle se trouve en 38912+8*A en mode HIRES.

— Chaque définition de caractère occupe 8 octets. Comme un code ASCII est compris entre Ø et 127, on a donc à priori 128×8=1Ø24=1 Koctet occupé par chacun des jeux de caractères. Mais en réalité, il n'y a pas de caractères correspondant aux codes Ø à 31. Les 32×8=256 premiers octets des deux zones sont donc inutilisés.

h) La zone écran

Elle est, bien entendu, plus grande en mode HIRES qu'en mode TEXT. En mode texte, on a 28 lignes de 40 caractères, comme un caractère occupe un octet mémoire, l'écran en mode TEXT utilise 28×4Ø=112Ø octets, des adresses 48ØØØ à 49119. Il est à noter que la première ligne de l'écran, celle où est située le mot CAPS correspond aux octets d'adresses 48ØØØ à 48Ø39, et peut-être accédée grâce à PEEK, DEEK, POKE et DOKE.

En mode HIRES, on a 2ØØ lignes de 24Ø points plus trois lignes de texte de 4Ø caractères. Comme chaque octet correspond à six points, une ligne HIRES utilise 24Ø/6=4Ø octets.

L'écran HIRES utilise donc en tout (2ØØ+3)×4Ø=812Ø octets, qui sont situés aux adresses 4Ø96Ø à 49Ø79.

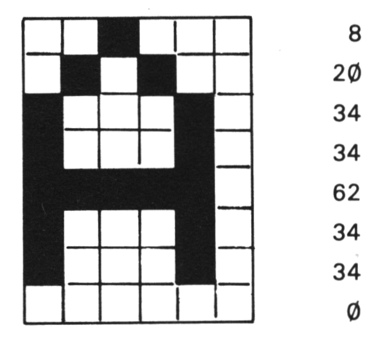
i) La ROM #CØØØ à #FFFF

On y trouve l'interpréteur Basic, appelé aussi plus simplement le Basic. Il est impossible d'écrire dans cette mémoire qui, par construc-

tion, ne peut être que lue. Par construction du microprocesseur 65Ø2, on trouve en FFFE, FFFC et FFFA les adresses où il se branche en cas de irq (interrupt request), de reset et de nmi (non masquable interrupt) respectivement. Ainsi, CALL DEEK(#FFFA) produit le même effet que lorsque l'on enfonce la touche "reset" et CALL DEEK(#FFFC) produit le même effet que lorsque l'on branche l'ORIC.

2.2 COMMENT REDÉFINIR LES CARACTÈRES

Chaque caractère, normal ou semi-graphique, est formé dans une matrice de 6 colonnes × 8 lignes.



Chaque caractère est codé par 8 octets contigus en mémoire, il y a un octet par ligne de la matrice. Le caractère "A" ci-dessus est codé par les 8 octets suivants :

Ø	Ø	Ø	Ø	1	Ø	Ø	Ø	Ø+ Ø+8+Ø+Ø+Ø= 8
Ø	Ø	Ø	1	Ø	1	Ø	Ø	Ø+16+Ø+4+Ø+Ø=2Ø
Ø	Ø	1	Ø	Ø	Ø	1	Ø	32+ Ø+Ø+Ø+2+Ø=34
Ø	Ø	1	Ø	Ø	Ø	1	Ø	32+ Ø+Ø+Ø+2+Ø=34
Ø	Ø	1	1	1	1	1	Ø	32+16+8+4+2+Ø=62
Ø	Ø	1	Ø	Ø	Ø	1	Ø	32+ Ø+Ø+Ø+2+Ø=34
Ø	Ø	1	Ø	Ø	Ø	1	Ø	32+ Ø+Ø+Ø+2+Ø=34
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø+ Ø+Ø+Ø+Ø+Ø= Ø
		32	16	8	4	2	1	

c'est-à-dire qu'à chaque point du caractère correspond un "1". Les deux bits de gauche de chaque octet sont inutilisés et valent Ø.

Les paquets de huit octets correspondant aux caractères sont rangés dans la mémoire en suivant l'ordre des codes ASCII des caractères. Par exemple, la zone de stockage des caractères standards débute en 46Ø8Ø en mode TEXT. Le premier des huit octets codant la lettre "A", dont le code ASCII est 65, est à l'adresse 46Ø8Ø+8*65=466ØØ. On trouve donc:

```
 8 en 466ØØ
2Ø en 466Ø1
34 en 466Ø2
34 en 466Ø3
62 en 466Ø4
34 en 466Ø5
34 en 466Ø6
 Ø en 466Ø7
```

Le premier des huit octets codant la lettre "B", de code ASCII 66, se trouve lui à l'adresse 46Ø8Ø+8*66=466Ø8.

Le programme:

```
1Ø INPUT "Donner un caractère";CAR$
2Ø TEXT:S=46Ø80+8*ASC(CAR$)
3Ø FOR I=S TO S+7
4Ø PRINT I,PEEK(I)
5Ø NEXT I
```

vous permet d'afficher les octets définissant un caractère standard.

Le programme:

```
1Ø S=46Ø8Ø+8*ASC("A"):TEXT
2Ø FOR I=S TO S+7
3Ø POKE I+8,PEEK(I)
4Ø NEXT I
```

recopie la définition du caractère "A" dans celle de "B". Après avoir exécuté ce programme, un "A" apparaîtra sur l'écran lorsque vous frapperez la touche "B", car le caractère de code ASCII 66 a maintenant la forme du "A" au lieu du "B".

Comme exemple, nous allons transformer le caractère @ en π.

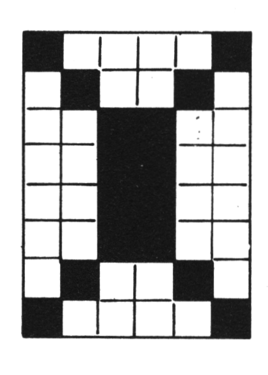
Ø	Ø	Ø	Ø	Ø	Ø	**1**	Ø	Ø+ Ø+Ø+Ø+2+Ø= 2
Ø	Ø	**1**	**1**	**1**	**1**	Ø	Ø	32+16+8+4+Ø+Ø=6Ø
Ø	Ø	Ø	**1**	Ø	**1**	Ø	Ø	Ø+16+Ø+4+Ø+Ø=2Ø
Ø	Ø	Ø	**1**	Ø	**1**	Ø	Ø	Ø+16+Ø+4+Ø+Ø=2Ø
Ø	Ø	Ø	**1**	Ø	**1**	Ø	Ø	Ø+16+Ø+4+Ø+Ø=2Ø
Ø	Ø	Ø	**1**	Ø	**1**	Ø	Ø	Ø+16+Ø+4+Ø+Ø=2Ø
Ø	Ø	Ø	**1**	Ø	**1**	Ø	Ø	Ø+16+Ø+4+Ø+Ø=2Ø
Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø+ Ø+Ø+Ø+Ø+Ø= Ø
		32	16	8	4	2	1	

En exécutant le programme:

```
1Ø TEXT:S=46Ø8Ø+ASC("@")*8
2Ø FOR I=S TO S+7:READ V:POKE I,V:NEXT
3Ø DATA 2,6Ø,2Ø,2Ø,2Ø,2Ø,2Ø,Ø
```

on redéfinit @. Chaque fois que vous frapperez la touche "@", un "π" est affiché.

Le petit insecte du jeu GLOUTON est un caractère redéfini:



Ø	Ø	**1**	Ø	Ø	Ø	Ø	**1**	32+ Ø+Ø+Ø+Ø+1=33
Ø	Ø	Ø	**1**	Ø	Ø	**1**	Ø	Ø+16+Ø+Ø+2+Ø=18
Ø	Ø	Ø	Ø	**1**	**1**	Ø	Ø	Ø+ Ø+8+4+Ø+Ø=12
Ø	Ø	Ø	Ø	**1**	**1**	Ø	Ø	Ø+ Ø+8+4+Ø+Ø=12
Ø	Ø	Ø	Ø	**1**	**1**	Ø	Ø	Ø+ Ø+8+4+Ø+Ø=12
Ø	Ø	Ø	Ø	**1**	**1**	Ø	Ø	Ø+ Ø+8+4+Ø+Ø=12
Ø	Ø	Ø	**1**	Ø	Ø	**1**	Ø	Ø+16+Ø+Ø+2+Ø=18
Ø	Ø	**1**	Ø	Ø	Ø	Ø	**1**	32+Ø+Ø+Ø+Ø+Ø+1=33
		32	16	8	4	2	1	

```
1Ø TEXT:S=46Ø8Ø+ASC("@")*8
2Ø FOR I=S TO S+7:READ V:POKE I,V:NEXT
3Ø DATA 33,18,12,12,12,12,18,33
```

redéfinit le @ en glouton.

On peut, exactement de la même façon, redéfinir les caractères semi-graphiques. En mode TEXT, ces caractères sont stockés à partir de l'adresse 47104, le premier octet d'un caractère semi-graphique de code ASCII a se trouve à l'adresse 47104+8*a.

Si vous désirez redéfinir un caractère tout en étant en mode HIRES, il faut utiliser l'adresse 38912 au lieu de 46080 pour les caractères standards, et 39936 au lieu de 47104 pour les caractères semi-graphiques. En effet, l'ORIC déplace dans sa mémoire les définitions des deux jeux de caractère à chaque fois que l'on passe d'un mode graphique à l'autre (TEXT et HIRES).

Pour retrouver le jeu de caractères normal, vous pouvez enfoncer la touche reset, ou exécuter CALL DEEK(#FFFA), qui a la même action que reset. Vous pouvez aussi exécuter CALL #F8D0 qui a le seul effet de recharger les caractères initiaux.

Pour terminer, voici un programme qui vous permettra de redéfinir n'importe quel caractère.

```
10 REM REDEFINITION DES CARACTERES***
20 :
30 TEXT:CLS:PAPER 2:INK 0:PRINT:PRINT
40 PRINT"Donnez le code ASCII ";
50 PRINT"(32-127) ";
60 INPUT COD%
70 IF COD%<32 OR COD%>127 THEN 40
80 PRINT"Quel jeu ";
90 PRINT"(0:normal,1:graphique) ";
100 INPUT JEU%
110 IF JEU%<0 OR JEU%>1 THEN 80
120 AD=46080+1024*JEU%+8*COD%
130 :
140 REM MODE D'EMPLOI----------------
150 CLS:PRINT
160 PRINT"Dessinez le caractere ";
170 PRINT"dans la grille"
180 PRINT:PRINT"Pour marquer un point:"
190 PRINT"-On deplace le curseur"
200 PRINT" avec les fleches"
210 PRINT"-On frappe X"
```

```
220 PRINT
230 PRINT"Pour effacer un point:"
240 PRINT"-On amene le curseur"
250 PRINT" sur le point"
260 PRINT"-On frappe la barre"
270 PRINT
280 PRINT"Lorsque le dessin est fini"
290 PRINT"frappez la touche ESC"
300 :
310 REM DESSINE LA GRILLE------------
320 FOR C=16 TO 23:PLOT C,16,"*":NEXT
330 FOR L=17 TO 24:PLOT 16,L,"*"
340 PLOT 23,L,"*":NEXT
350 FOR C=16 TO 23:PLOT C,25,"*":NEXT
360 :
370 REM PLACE CURSEUR DANS GRILLE----
380 PRINT:PRINT:FOR I=1 TO 15
390 PRINT CHR$(9);
400 NEXT:X=1:Y=1
410 :
420 REM DESSIN DU CARACTERE----------
430 GET A$:IF A$=CHR$(27) THEN 540
440 IF A$="X"ANDX<7THEN X=X+1:GOTO520
450 IF A$=" "ANDX<7THEN X=X+1:GOTO520
460 A=ASC(A$):IF A<8 OR A>11 THEN 420
470 IF A=8 AND X>1 THEN X=X-1:GOTO520
480 IF A=9 AND X<6 THEN X=X+1:GOTO520
490 IF A=10AND Y<8 THEN Y=Y+1:GOTO520
500 IF A=11AND Y>1 THEN Y=Y-1:GOTO520
510 GOTO 420
520 PRINT A$;:GOTO 420
530 :
540 REM RANGE CETTE DEFINITION DANS T
550 DIM T(7):PAPER 1
560 FOR I=0 TO 7
570 FOR C=17 TO 22
580 IF SCRN(C,17+I)<>ASC("X") THEN600
590 T(I)=T(I)+2^(22-C)
600 NEXT C
610 NEXT I:PAPER 2:CLS
620 :
```

```
630 REM RANGE EN MEMOIRE ORIC--------
640 PRINT:PRINT
650 FOR I=0 TO 7
660 POKE AD+I,T(I)
670 PRINT"POKE ";AD+I;",";T(I)
680 NEXT I
690 PRINT:PRINT"On obtient:";
700 PRINT CHR$(27);CHR$(72+JEU%);
710 PRINT CHR$(COD%)
720 :
730 REM ENCORE ?---------------------
740 PRINT:PRINT
750 PRINT"Encore (O/N) ";:GET A$
760 IF A$="O" THEN RUN
770 IF A$="N" THEN END
780 PING:GOTO 750
```

2.3 TRUCS ET FICELLES DES MODES TEXT ET LORES

En mode TEXT ou LORES, l'écran est divisé en 28 lignes de 4Ø caractères chacune. A chacun de ces 4Ø×28=112Ø caractères correspond un octet bien particulier de la mémoire de l'ORIC. Par exemple, le premier caractère de la première ligne correspond à l'octet 48ØØØ (#BB8Ø en hexadécimal). Pour faire apparaître un caractère sur l'écran, l'ORIC place son code ASCII dans l'octet correspondant. Vous pouvez faire de même avec la fonction POKE. Par exemple, pour placer la lettre "A", dont le code ASCII est 65, dans le premier caractère de la première ligne, exécutez :

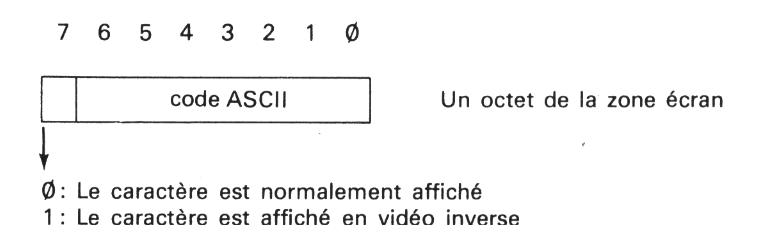
```
POKE 48ØØØ,65
```

Si l'on numérote les lignes de Ø à 27, la ligne Ø étant la ligne supérieure, et les caractères de Ø à 39, l'adresse de l'octet correspondant au caractère X de la ligne Y vaut 48ØØØ+4Ø*Y+X.

De ce qui précède, on peut déduire un premier "truc", qui permet d'écrire sur la première ligne de l'écran, celle où se trouve le mot CAPS.

```
 1Ø A$="Et voila le travail!!"
 2Ø GOSUB 1ØØ
 3Ø END
 4Ø :
1ØØ REM ECRITURE A$ SUR LA 1 LIGNE
11Ø FOR I=1 TO LEN(A$)
12Ø POKE 47999+I,ASC(MID$(A$,I,1))
13Ø NEXT I
14Ø RETURN
```

Comme on l'a dit, à chaque position de l'écran correspond un octet en mémoire, pour faire apparaître un caractère à un endroit déterminé, on place son code ASCII dans l'octet correspondant. Or un code ASCII est un nombre compris entre Ø et 127, qui s'exprime sur 7 bits. Comme un octet contient 8 bits, il reste donc a priori un bit inutilisé, le bit de poids fort. En fait, ce bit est utilisé lorsque l'on affiche le caractère en vidéo inverse.



Pour faire passer à "1" le bit de poids fort, il suffit d'ajouter 1ØØØØØØØ (binaire) au code ASCII, ce qui vaut encore 128 en base dix ou 8Ø en hexadécimal.

Ceci va nous permettre d'écrire en vidéo inverse, tout d'abord en reprenant le petit programme précédent :

```
 1Ø A$="Et voila le travail"
 2Ø GOSUB 1ØØ
 3Ø END
 4Ø :
1ØØ REM ECRITURE A$ SUR LA 1 LIGNE
11Ø FOR I=1 TO LEN(A$)
12Ø POKE 47999+I,ASC(MID$(A$,I,1))+128
13Ø NEXT I
14Ø RETURN
```

On a simplement ajouté +128 à la ligne 12Ø.

On peut aussi utiliser la fonction PLOT pour écrire en vidéo inverse, comme dans l'exemple suivant :

```
 1Ø TEXT:CLS
 2Ø X=9:Y=1Ø:A$="Ceci est un message"
 3Ø GOSUB 1ØØ
 4Ø X=14:Y=13:A$="ecrit en"
 5Ø GOSUB 1ØØ
 6Ø X=12:Y=16:A$="video inverse"
 7Ø GOSUB 1ØØ
 8Ø WAIT 5Ø:GOTO 1Ø
 9Ø :
1ØØ REM PLOT EN VIDEO INVERSE
11Ø MUSIC 1,4,1,Ø
12Ø FOR I=1 TO LEN(A$)
13Ø PLOT X+I−1,Y,ASC(MID$(A$,I,1))+128
14Ø PLAY 1,Ø,Ø,25:WAIT 15
15Ø NEXT I
16Ø PING:WAIT 5Ø
17Ø RETURN
```

L'affichage se fait à la ligne 13Ø. MID$(A$, I, 1) donne le caractère I de la chaîne A$. ASC(MID$(A$, I, 1)) donne le code ASCII de ce caractère, auquel on ajoute 128 pour obtenir un effet de vidéo inverse.

Les codes ASCII Ø à 31 ont un rôle un peu particulier. Lorsqu'on les place dans un octet de la zone écran, aucun caractère n'apparaît. En revanche, la ligne où ils se trouvent change d'aspect, à partir de l'endroit ils se trouvent. Ils permettent de choisir la couleur de l'encre, de l'écran, de sélectionner le jeu de caractère, la taille des caractères et de faire clignoter des caractères. Comme ils modifient l'aspect d'une ligne où ils se trouvent, on les appelle parfois "attributs". L'annexe 3.2 donne la liste de ces caractères.

Par exemple, 18 donne à l'écran la couleur verte. Tapez POKE 48Ø2Ø, 18.

L'octet 48Ø2Ø correspond au milieu de la première ligne de l'écran qui maintenant est verte. Si vous tapez maintenant :

```
POKE 48Ø2Ø,17
```

on obtient du rouge. Tapez :

`POKE 48Ø2Ø,12` et le mot CAPS se met à clignoter.

Ces codes peuvent aussi être employés avec PLOT, comme le montre l'exemple suivant :

```
 1Ø TEXT:PAPER Ø:INK 1:CLS
 2Ø PLOT 11,1,CHR$(14)+"DEMONSTRATION"
 3Ø PLOT 11,2,CHR$(14)+"DEMONSTRATION"
 4Ø :
 5Ø PLOT 15,12,CHR$(1)+"ROUGE"
 6Ø PLOT 15,13,CHR$(2)+"VERT"
 7Ø PLOT 15,14,CHR$(3)+"JAUNE"
 8Ø PLOT 15,15,CHR$(4)+"BLEU"
 9Ø PLOT 15,16,CHR$(5)+"MAUVE"
1ØØ PLOT 15,17,CHR$(6)+"CYAN"
11Ø PLOT 15,18,CHR$(7)+"BLANC"
12Ø :
13Ø FOR I=1 TO 7:WAIT 2ØØ:PING
14Ø FOR J=11 TO 19
15Ø PLOT 14,J,I+16:PLOT 23,J,16
16Ø NEXT J
17Ø NEXT I
```

Le code 14 donne des lettres clignotantes double hauteur, les codes 1 à 7 fixent la couleur de l'encre. A la ligne 15Ø, I+16 varie de 17 à 23 et fixe la couleur de l'écran.

Pour afficher ces codes Ø à 31 avec un ordre PRINT, il faut exécuter PRINT CHR$(27); CHR$(code+64), voir l'annexe 3.2.

En effet, si vous exécutez PRINT CHR$(code) et que le code est compris entre Ø et 31, l'ORIC **n'affiche pas** ce code. A la place, il exécute un sous-programme de sa mémoire ROM. Par exemple, si vous exécutez PRINT CHR$(12), l'ORIC ne placera pas ce code dans la mémoire-écran, ne l'affichera pas, mais exécutera un sous-programme qui efface l'écran, en remplissant la mémoire-écran de codes 32 (blanc). L'effet de PRINT CHR$(code) pour les valeurs Ø à 31 est décrit au début de l'annexe 3.1.

Voici un petit programme démontrant quelques-unes des possibilités de ces codes avec PRINT, que l'on appelle caractères de contrôle, car ils permettent en majorité de contrôler le curseur.

```
1Ø PRINT CHR$(12):REM EFFACE ECRAN
2Ø :
3Ø PRINT @15,26;"COUCOU"
4Ø FOR I=1 TO 26
```

```
 5Ø FOR I=1 TO 26
 6Ø PRINT CHR$(1Ø); :REM VERS LE BAS
 7Ø WAIT 2Ø
 8Ø NEXT
 9Ø :
1ØØ PRINT CHR$(3Ø); :REM RETOUR
11Ø :
12Ø FOR I=26 TO 2 STEP -2
13Ø FOR J=1 TO I :PRINT CHR$(9); :NEXT
14Ø FOR J=1 TO I :PRINT CHR$(1Ø); :NEXT
15Ø FOR J=1 TO I-1 :PRINT CHR$(8); :NEXT
16Ø FOR J=1 TO I-1 :PRINT CHR$(11); :NEXT
17Ø NEXT I
```

Les codes 8, 9, 1Ø et 11 sont les codes ASCII des touches flèches à gauche, à droite, en bas et en haut. On peut grâce à eux déplacer le curseur, à condition de ne pas oublier le ";" à la fin de l'ordre PRINT. Ce ";" empêche PRINT d'envoyer le curseur au début de la ligne suivante.

Pour terminer, voici quelques adresses qui vous assureront la maîtrise totale de l'écran de l'ORIC.

18 ou #12 19 ou #13	:	adresse-mémoire du premier octet de la ligne où se trouve le curseur.
616 ou #268	:	numéro de la ligne où est le curseur (Ø à 27.
617 ou #269	:	numéro de la colonne où est le curseur (Ø à 39).
618 ou #26A	:	bascules CTRL-D, CTRL-], CTRL-[, CTRL-F, CTRL-S et CTRL-Q.
619 ou #26B	:	code pour PAPER (16 à 23).
62Ø ou #26C	:	code pour INK (Ø à 7).
632 ou #278 633 ou #279	:	adresse-mémoire du premier octet de la seconde ligne de l'écran (#BBDØ en TEXT, #BF9Ø en HIRES).
634 ou #27A 635 ou #27B	:	adresse-mémoire du premier octet de la première ligne de l'écran (#BBA8 en TEXT, #BF68 en HIRES).
636 ou #27C 637 ou #27D	:	nombre de caractères à enrouler (26×4Ø=1Ø4Ø en TEXT, 2×4Ø=8Ø en HIRES).
638 ou #27E	:	nombre de lignes de l'écran (27 en TEXT ou 3 en HIRES).

Ces adresses sont à séparer en quatre groupes :

— Celles qui servent au curseur : 18, 19, 616 et 617.

Le curseur est en réalité un caractère qui est affiché normalement, puis en vidéo inverse, puis normalement, à nouveau en vidéo inverse, etc... L'adresse-mémoire du premier octet de la ligne où se trouve le curseur est stockée en 18-19.

En 616 est stocké le numéro de la ligne courante, où sera affichée le prochain caractère. Normalement, ceci correspond à la ligne où se trouve le curseur. En 617 est stocké le numéro de la colonne où se trouve le curseur.

Cette adresse peut servir à remplacer la fonction TAB. Tapez:

```
POKE 617,1Ø:PRINT "COUCOU"
```

Nous allons maintenant utiliser les adresses 18, 19 et 616 pour simuler une instruction PRINT X, Y, A$. Cette instruction a le même effet que PLOT @ X, Y; A$, mais peut écrire sur la première ligne de l'écran (Y=Ø) et dans les deux premières colonnes (X=Ø ou 1). Considérons le programme suivant:

```
10 PRINT CHR$(30);:INPUT"X,Y ";X,Y
20 INPUT"CHAINE ";A$
30 GOSUB 100
40 GET A$:IF A$<>"A" THEN 10
50 END
60 :
100 REM PRINT AT X,Y,A$--------------
110 L=DEEK(18):ACUR=L+PEEK(617)
115 DOKE £400,£6078:CALL £400
120 CAR=PEEK(ACUR)
130 IF CAR>127 THEN POKE ACUR,CAR-128
140 L=£BB80+Y*40
150 POKE 616,Y:POKE 617,X
160 DOKE 18,L
165 DOKE £400,£6058:CALL £400
170 PRINT A$
180 RETURN
```

Note: le caractère "£" est à remplacer par "#".

Nous allons expliquer le fonctionnement du sous-programme 1ØØ. Comme on l'a dit, le curseur est en fait un caractère que l'ORIC fait passer en vidéo inverse, en vidéo normale, en vidéo inverse, etc... Comme ce sous-programme va déplacer le curseur, pour le mettre à la position

X, Y, la première chose à faire est d'effacer le curseur de l'endroit où il se trouve, ce qui est fait par les lignes 11Ø à 13Ø.

11Ø On place dans ACUR l'adresse-mémoire de l'octet sur lequel se trouve le curseur.

115 Nous laissons de côté cette ligne pour l'instant.

12Ø On place dans CAR la valeur de l'octet sur lequel est le curseur.

13Ø Si CAR est supérieur à 127, il est affiché en vidéo inverse et on le fait passer en vidéo normale.

De 14Ø à 16Ø, on place le curseur en X, Y.

14Ø BB8Ø est l'adresse du début de l'écran. Comme une ligne comporte 4Ø caractères, ACUR contient maintenant l'adresse du nouvel octet sur lequel on désire placer le curseur.

15Ø On place en 616 le numéro Y de la ligne, en 617 le numéro de la colonne.

16Ø On place en 18 la nouvelle adresse de la ligne où est le curseur.

17Ø On imprime la chaîne A$.

Nous allons maintenant nous pencher sur les lignes 115 et 165. Pour faire clignoter le curseur, l'ORIC agit ainsi : A l'intérieur de l'ORIC se trouve une sorte de réveil-matin, appelé TIMER, qui sonne toutes les demi-secondes environ. A chaque fois que le réveil sonne, l'ORIC laisse tomber purement et simplement ce qu'il était en train de faire, exécuter votre programme Basic ou attendre que vous frappiez quelque chose, pour se précipiter en 18-19. Il lit le nombre qui se trouve en 18-19, lui ajoute le contenu de 617, et va à cette adresse. Là, il trouve l'octet sur lequel est le curseur. Si cet octet est supérieur à 128, il donne un caractère en vidéo inverse, alors l'ORIC retranche 128 à cet octet pour obtenir le même caractère, mais en vidéo normale. Au contraire, si l'octet est inférieur à 128, l'ORIC lui ajoute 128 pour obtenir l'effet de vidéo inverse. Ceci fait, l'ORIC retourne au travail qu'il avait abandonné. Et cela recommence toutes les demi-secondes, l'ORIC fait passer en vidéo inverse, puis normale, puis inverse, etc... l'octet dont l'adresse est stockée en 18-19 et 617.

Maintenant, supposons que le réveil sonne pendant que l'ORIC exécute la ligne 14Ø ou 15Ø. A la ligne 13Ø, on avait veillé à faire passer le caractère sur lequel est le curseur en vidéo normale. Puis, par exemple,

vers le milieu de la ligne 14Ø, le réveil retentit. Aussitôt, l'ORIC laisse tomber mon programme et va faire passer en vidéo inverse ce caractère, dont l'adresse est en 18-19 et 617. Puis il revient à mon programme, qui à la ligne 16Ø modifie le contenu de 18-19. Lors de la prochaine sonnerie, l'ORIC ira donc modifier l'octet dont l'adresse et la nouvelle valeur sont en 18-19 et 617. Ainsi, le caractère dont l'adresse était l'ancienne valeur stockée en 18-19 et 617 se trouve toujours, le pauvre, en vidéo inverse. On a donc sur l'écran un caractère qui reste en vidéo inverse, du plus mauvais effet. Pour parer à cela, c'est très simple, il suffit de rendre l'ORIC sourd à la sonnerie, ce qui est fait en 115. En effet, on place en #4ØØ un #78, qui est le code d'une instruction SEI et en #4Ø1 on met #6Ø, RTS. Puis on exécute ce sous-programme machine de deux instructions. C'est l'instruction SEI qui rend sourd l'ORIC. Puis en 165, on lui rend l'ouïe, en exécutant le sous-programme constitué de #58 et #6Ø, soit CLI et RTS. L'instruction CLI à l'effet inverse de SEI.

Si vous voulez mieux voir ce qui se passe, ajoutez la ligne:

```
117 FOR I=1 TO 4ØØØ:NEXT I
```

Puis tapez:

CLS:RUN return	L'ORIC vous demande:
X,Y?	Tapez:
1Ø,1Ø return	L'ORIC demande alors:
CHAINE?	Tapez:
ABCD return	

Vous observez alors que pendant que la ligne 117 s'exécute, le curseur se trouve au début de la troisième ligne, **et qu'il ne clignote plus**. En effet, la ligne 115 a masqué les interruptions, a rendu sourd l'ORIC, qui ne va plus faire changer d'état son curseur.

— Nous passons maintenant au second groupe, les adresses 632 à 638 qui servent à définir la taille de l'écran pour PRINT. On trouve en 632-633 l'adresse de la seconde ligne, qui vaut normalement #BBDØ en mode TEXT et #BF9Ø en mode HIRES. En 634-635 se trouve l'adresse de la première ligne, #BBA8 en TEXT et #BF68 en HIRES. Le nombre de caractères à déplacer lorsque l'écran est plein, à enrouler, se trouve en 636-637, et vaut 26×4Ø=1Ø4Ø en TEXT, 2×4Ø=8Ø en HIRES. Enfin, en 638 se trouve le nombre de lignes de l'écran (27 en TEXT, 5 en HIRES).

Les ordres PRINT, CLS utilisent la zone dont l'adresse de début vaut DEEK(634), et comportant PEEK(638) lignes de 4Ø caractères. En modifiant ces valeurs, vous pouvez modifier la partie utilisable de l'écran. Tapez :

```
DOKE 636,7*4Ø:POKE 638,8:CLS return
```

Ceci restreint à huit lignes la zone utilisable de l'écran. Déplacez le curseur avec flèche en bas : il ne peut dépasser la 8e ligne.

Tapez maintenant :

```
DOKE 632,#BB8Ø+4Ø:DOKE 634,#BB8Ø:DOKE 636,27*4Ø:
POKE 638,28:CLS return
```

PRINT peut maintenant écrire depuis #BB8Ø sur 28 lignes. C'est-à-dire que vous pouvez maintenant utiliser la ligne du haut de l'écran, celle où se trouve CAPS. Tapez et exécutez :

```
 1Ø DOKE 632,#BB8Ø+4Ø:DOKE 634,#BB8Ø
 2Ø DOKE 636,27*4Ø:POKE 638,28:CLS
 3Ø PRINT SPC(12);"ZONE PROTEGEE"
 4Ø FOR I=#BB8Ø+4Ø*5 TO#BB8Ø+4Ø*5+39
 5Ø POKE I,ASC("=")
 6Ø NEXT I
 7Ø :
 8Ø DOKE 632,#BB8Ø+7*4Ø
 9Ø DOKE 634,#BB8Ø+6*4Ø
1ØØ DOKE 636,21*4Ø:POKE 638,22
11Ø CLS
```

Vous pouvez déplacer le curseur, exécuter CLS ou CTRL-L sans atteindre les deux lignes supérieures de l'écran. Les lignes 1Ø-2Ø définissent un écran de 28 lignes, tandis que 8Ø-11Ø définissent un écran de 22 lignes.

— Le troisième groupe d'adresses est formé de 619 et 62Ø. On y trouve respectivement les attributs utilisés pour définir la couleur de l'écran et de l'encre lorsque CLS ou CTRL-L est exécuté. Ces deux commandes agissent comme suit : Pour chaque ligne, on place dans le premier octet la valeur qui est en 619, on place dans le second octet la valeur qui est en 62Ø, et on met un code 32 (blanc) dans les 38 octets restants de la ligne.

Reprenons le petit programme précédent, on ajoute les lignes 25 et 12Ø pour obtenir :

```
10 DOKE 632,#BB80+40:DOKE 634,#BB80
20 DOKE 636,27*40:POKE 638,28:CLS
25 POKE 619,16:POKE 620,7:CLS
30 PRINT SPC(12);"ZONE PROTEGEE"
40 FOR I=#BB80+40*5 TO # BB80+40*5+39
50 POKE I,ASC('=')
60 NEXT I
70 :
80 DOKE 632,#BB80+7*40
90 DOKE 634,#BB80+6*40
100 DOKE 636,21*40:POKE 638,22
110 CLS
120 POKE 619,18: POKE 620,0: CLS
```

En 619, on doit placer un attribut qui détermine la couleur de l'écran, c'est-à-dire un nombre de 16 à 23, tandis qu'en 620 on met un attribut qui détermine la couleur de l'encre, un nombre compris entre 0 et 7. A la ligne 20, on choisit un écran noir (16 en 619) avec de l'encre blanche (7 en 620). A la ligne 90, on fixe un écran vert (18 en 619) et de l'encre noire (0 en 620).

— Pour terminer, le quatrième groupe qui ne comporte qu'une adresse 618, les 7 bits de poids faible de cet octet ont chacun une signification particulière, que nous donnons dans le tableau ci-dessous.

Numéro du bit	7	6	5	4	3	2	1	0
Poids du bit	128	64	32	16	8	4	2	1
action du bit à 0	?	Affichage simple	Affichage sur 38 colonnes	pas de ESC	clavier sonore	?	plus d'affichage	curseur pas affiché
Action du bit à 1	?	Affichage double	Affichage sur 40 colonnes	ESC a été détecté	clavier silencieux	?	affichage	curseur affiché
Touche faisant changer d'état le bit	aucune	CTRL-D	CTRL-]	CTRL-Z ESC	CTRL-F	CTRL-P	CTRL-S	CTRL-Q
Caractère de contrôle changeant d'état le bit	aucun	CHR$(4)	CHR$(29)	CHR$(26) CHR$(27)	CHR$(6)	CHR$(16)	CHR$(19)	CHR$(17)

Lorsque l'ORIC est branché, ou lorsque l'on enfonce "reset", l'ORIC place en 618 la valeur :

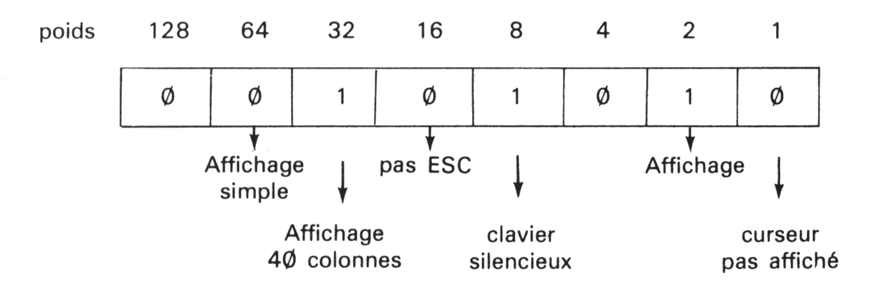
618 :	Ø	Ø	Ø	Ø	Ø	Ø	1	1

Ce qui vaut Ø+Ø+Ø+Ø+Ø+2+1=3. Ceci correspond à :

* Affichage simple
* Affichage sur 38 colonnes
* La touche précédente n'était pas ESC
* Le clavier est sonore
* Les caractères sont affichés
* Le curseur est affiché.

Les touches CTRL et les caractères de contrôle font changer d'état l'un de ces bits, s'il était à Ø, on obtient 1, et réciproquement. En exécutant un ordre POKE à l'adresse 618, vous pouvez initialiser tous ces bits aux valeurs de votre choix. Par exemple, après avoir frappé un nombre quelconque de touches de contrôle, POKE 618,3 vous ramène bien dans l'état auquel vous êtes habitué, celui de la mise en marche.

Si vous désirez maintenant être en affichage simple, sur 4Ø colonnes, avec un clavier silencieux et que le curseur ne soit pas affiché, il faut placer en 618 la valeur :



Ce qui vaut Ø+Ø+32+Ø+8+Ø+2+Ø=42. On exécute donc :

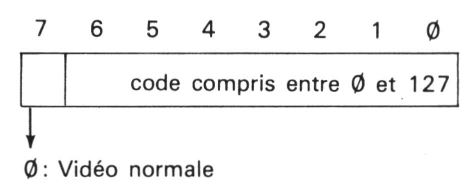
```
POKE 618,42
```

2.4 TRUCS ET FICELLES DU MODE HIRES

L'écran HIRES est divisé en 2Ø3 lignes de 4Ø caractères chacune, 2ØØ lignes de graphique suivies de trois lignes de texte. Exactement comme en mode texte, à chacun de ces 2Ø3×4Ø=812Ø caractères correspond un octet de la mémoire de l'ORIC, de 4Ø96Ø à 49Ø79 (#AØØØ à #BFB7).

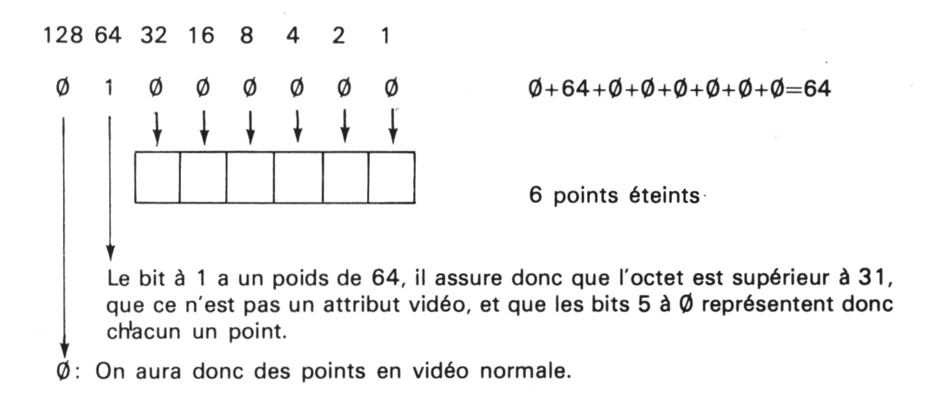
Si l'on numérote les lignes graphiques de Ø à 199, la ligne Ø étant la ligne supérieure, et les caractères de Ø à 39, l'adresse de l'octet correspondant au caractère X de la ligne Y vaut 4Ø96Ø+4Ø*Y+X.

Comme en mode TEXT, une ligne est constituée de 4Ø octets, et le bit de poids fort de chacun de ses octets sélectionne le mode vidéo normale ou inverse.

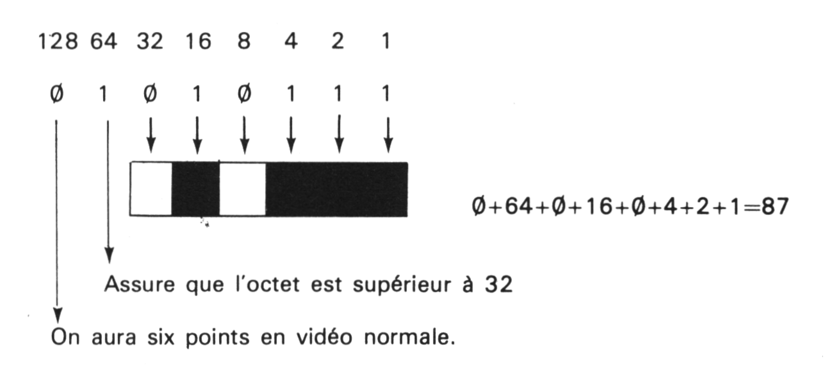


Les 7 bits restants peuvent donner un nombre compris entre Ø et 127. Comme en mode TEXT, les valeurs Ø à 31 sont des attributs vidéos (voir annexe 3.2). Par contre, les valeurs 32 à 127 ne sont pas des codes ASCII de caractères. A la place, chacun des six bits 5 à Ø correspond à un point de l'écran. Ainsi, tout octet compris entre 32 et 127 donne six points. Comme on a 4Ø caractères par ligne, on peut donc avoir 6×4Ø=24Ø points par ligne. Chaque bit à zéro donne un point éteint, tandis qu'un bit à un donne un point allumé, visible.

Ainsi, l'octet :



correspond-il à six points éteints. Lorsque vous exécutez HIRES, l'ORIC place cette valeur, soit 64, dans les 2ØØ×4Ø octets qui constituent la zone graphique, on a donc un écran dont tous les points sont éteints. Prenons un second exemple, avec l'octet:



Cet octet vaut 87 et donne un point éteint, un point allumé, un point éteint et trois points allumés. Tapez:

```
HIRES return                 Puis:
POKE #AØØØ,87 return
```

#AØØØ est l'adresse du premier octet de la zone HIRES, on voit donc apparaître les six points correspondant à la valeur 87 dans le coin supérieur gauche de l'écran. Tapez maintenant:

```
FOR I=#AØØØ TO # AØ27:POKE I,87:NEXT return
```

Ceci pl'ace la valeur 87 dans les 4Ø premiers octets de la zone écran HIRES, qui correspondent à la première ligne de l'écran. Tapez maintenant :

```
FOR I=#AØØØ TO # BF3F:POKE I,87:NEXT return
```

Ceci remplit les 2ØØ lignes graphiques avec ce nombre 87.

Comme il n'est pas possible de placer dans un même octet à la fois une valeur inférieure à 32 et une valeur supérieure à 32, à chaque fois que vous placez un attribut vidéo dans la mémoire HIRES, les six points correspondant à cet octet seront toujours éteints, comme l'illustre l'exemple suivant. Tapez :

```
HIRES return
FOR I=#AFAØ TO# AFC7:POKE I,127:NEXT return
```

Ce qui trace une ligne au milieu de l'écran, 127 correspond à 7 bits à "1". Nous allons maintenant placer au milieu de cette ligne l'attribut "encre rouge" qui a pour code 1. Tapez :

```
POKE #AFB4,1 return
```

La fin de la ligne est bien rouge, et six points sont éteints. Tapez :

```
POKE #AFB4,127 return
```

Les six points sont revenus, mais la couleur rouge a disparu.

Lorsque l'on désire remplir une zone rectangulaire de la mémoire écran HIRES avec un code, l'ordre FILL est très pratique. Les paramètres de cet ordre sont la hauteur du rectangle en nombre de lignes, la largeur du rectangle en nombre d'octets et le code à placer dans toute cette zone.

```
FILL  nombre de lignes, nombre d'octets, code.
```

Avant d'employer FILL, il faut préciser à l'ORIC les coordonnées du coin supérieur gauche du rectangle, par exemple avec CURSET.

Exemples :

Tapez :

```
HIRES return
CURSET 6Ø,5Ø,3:FILL 1ØØ,2Ø,127 return
```

On place sur 1ØØ lignes×2Ø octets le code 127 qui correspond à six points allumés, on obtient donc un rectangle de 1ØØ×12Ø points. Tapez :

```
CURSET 54,5Ø,3:FILL 1ØØ,1,2 return
```

On place devant chacune des lignes du rectangle un code 2, qui est l'attribut "encre verte". Le rectangle est donc vert. Tapez maintenant :

```
CURSET 12Ø,5Ø,3:FILL 1ØØ,1,4 return
```

Ceci place 4, qui est l'attribut "encre bleue" au milieu du rectangle. Les points où sont les attributs sont éteints, tandis que les points à droite de ces attributs sont bleus.

Comme le montre l'exemple précédent, l'utilisation des attributs vidéo ne permettent pas d'avoir sur une même ligne des points voisins de couleurs différentes. Cela est possible en utilisant la vidéo inverse. Tapez :

```
HIRES return
CURSET 6Ø,5Ø,3:FILL 1ØØ,7,127 return
```

On obtient un rectangle blanc que nous allons colorier en vert en exécutant :

```
CURSET 54,5Ø,3:FILL 1ØØ,1,2 return
```

Tapez maintenant :

```
CURSET 1Ø2,5Ø,3:FILL 1ØØ,7,127+128 return
```

127+128=255 donne six points allumés en inverse vidéo.

Comme l'inverse vidéo du vert est le mauve, on obtient un rectangle mauve contigu au rectangle vert précédent. Tapez :

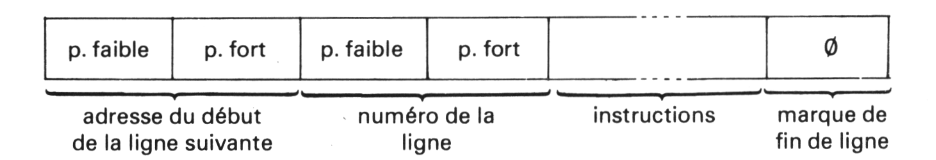
```
CURSET 144,5Ø,3:FILL 1ØØ,7,127 return
```

On obtient un nouveau rectangle vert contigu au rectangle mauve.

2.5 FABRIQUER DES LIGNES DE PROGRAMME PAR PROGRAMME

Lorsque l'on tape de longues listes de DATA, ou de grands textes après des ordres PRINT, il devient rapidement fastidieux de devoir à chaque ligne taper un numéro, le mot DATA ou PRINT. Il est préférable de ne taper que les données, et de laisser le soin à un programme d'ajouter les détails que sont le numéro de la ligne ou le mot DATA. Cela est possible lorsque l'on sait que :

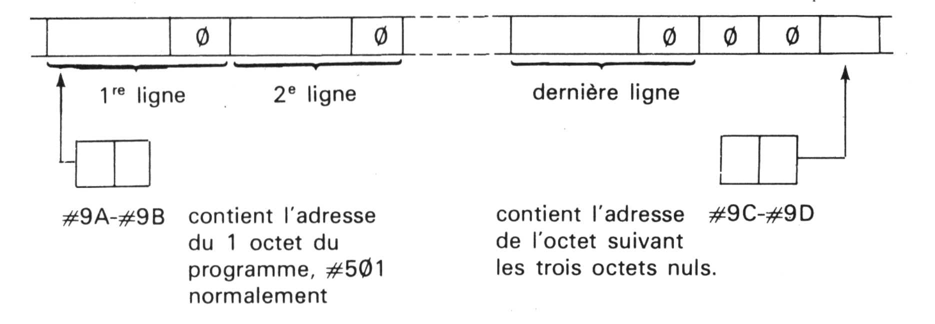
— Chaque ligne de programme est stockée en mémoire selon le format suivant :



On a d'abord sur deux octets l'adresse-mémoire de la ligne suivante. puis encore sur deux octets, le numéro de la ligne. Ensuite les caractères qui constituent les instructions, terminés par un octet nul qui indique la fin de la ligne.

Les instructions sont elles-mêmes stockées de façon simple : tous les mots réservés, comme ZAP, FOR, PRINT, etc... sont codés par un octet supérieur à 128. Tous les autres caractères (noms de variables, constantes numériques et chaînes de caractères) sont codés par leur code ASCII, compris entre 32 et 125.

— Les lignes sont classées en mémoire selon l'ordre de leurs numéros. Il n'y a pas de place perdue, de trou entre deux lignes successives. En #9A-#9B se trouve l'adresse du premier octet du programme, cette adresse vaut normalement #5Ø1. Juste après le Ø terminant la dernière ligne du programme, on trouve deux autres octets nuls, on a donc en tout trois octets nuls. En #9C-#9D se trouve l'adresse de l'octet qui suit immédiatement ces trois octets nuls.



Supposons que vous désiriez trouver l'adresse de début d'une ligne de programme, par exemple la ligne 5ØØ. Comme l'adresse de début de programme est stockée en #9A-#9B et que chaque ligne débute par l'adresse de la ligne suivante suivie de numéro, il suffit de taper en mode immédiat.

```
A=DEEK(#9A):N=5ØØ return
REPEAT:A=DEEK(A):UNTIL DEEK(A+2)=N return
PRINT A return
```

Les deux points qui précèdent nous suffisent par écrire un programme qui fabrique des lignes de DATA :

```
1 GOTO 1000
10 REM CES REM FIGURENT LE
20 REM PROGRAMME AUQUEL
30 REM ON DESIRE AJOUTER DES
40 REM ORDRES DATA.
50 :
1000 DOKE £9C,DEEK(£9C)+5000:CLEAR
1010 A=DEEK(£9C)-5000-2:N=2000
1020 L$="":Z=FRE(""):PRINT">";
1030 GET A$:IF A$=CHR$(13) THEN 1050
1040 L$=L$+A$:PRINT A$;:GOTO 1030
1050 IF L$="" THEN 1130
1060 DOKE A,A+LEN(L$)+6:DOKE A+2,N
1070 POKE A+4,145
1080 FOR I=1 TO LEN(L$)
```

```
1090 POKE A+4+I,ASC(MID$(L$,I,1))
1100 NEXT I
1110 POKE A+5+LEN(L$),0:A=A+LEN(L$)+6
1120 PRINT:N=N+10:GOTO 1020
1130 DOKE A,0:DOKE £9C,A+2:CLEAR
```

Note: le caractère "£" est à remplacer par "#".

Avant d'exécuter ce programme, *vérifiez soigneusement* que les lignes 1ØØØ à 113Ø sont bien correctes. A chaque fois que le programme affiche ">", tapez une ligne suivie de RETURN. Pour sortir du programme, tapez RETURN immédiatement après le ">".

Exemples :

```
RUN return
>1,2,3,4,5 return
>6,7,8,9,1Ø return
> return
Ready
LIST return
```

L'ORIC a ajouté les deux lignes :

```
2ØØØ DATA 1,2,3,4,5
2Ø1Ø DATA 6,7,8,9,1Ø
```

Vous pouvez changer le numéro de la première ligne, défini par l'instruction N=2ØØØ de la ligne 1Ø1Ø. On peut changer l'incrément des numéros de lignes, en modifiant l'instruction N=N+1Ø de la ligne 112Ø. Vous pouvez aussi ajouter la ligne :

```
1Ø35 IF A$=CHR$(24) THEN PRINT" \ ":GOTO 1Ø2Ø
```

qui, en cas d'erreur, permet d'annuler la ligne en cours de frappe en utilisant la touche CTRL-X.

Le bloc 1ØØØ à 113Ø fonctionne de la façon suivante : Le nombre stocké en #9C-#9D est l'adresse de la fin du programme Basic mais aussi celle du début de la zone des variables simples. Pour que les lignes de DATA que le programme va ajouter à sa suite ne détruisent pas ses variables, on commence donc, en 1ØØØ, par repousser de 5ØØØ l'adresse contenue en #9C. On dispose donc maintenant de 5ØØØ octets entre la fin du programme et le début des variables. L'ordre CLEAR qui suit

remet à jour les adresses stockées en #9E-#9F et #AØ-#A1 (début et fin de la zone de stockage des tableaux).

1Ø1Ø On va placer dans A l'adresse de la première ligne que l'on va ajouter. On met dans N le numéro de cette première ligne.

1Ø2Ø L$ va contenir la ligne que vous allez taper. Ne pas oublier le ";" à la fin de la ligne.

1Ø3Ø On lit un caractère. Si c'est RETURN, on va en 1Ø5Ø.

1Ø4Ø On ajoute le caractère lu à la chaîne A$ et on l'affiche. Ne pas oublier le ";" de l'ordre PRINT. Puis on va relire un autre caractère.

Lorsqu'un RETURN a été rencontré, on arrive en 1Ø5Ø.

1Ø5Ø Si la ligne est vide, on sort du programme, on va en 113Ø.

1Ø6Ø On place à l'adresse A, qui correspond au début de la ligne, l'adresse du début de la ligne suivante. Puis l'on place en A+2 le numéro de la ligne.

1Ø7Ø On place 145, qui est le code du mot réservé DATA.

1Ø8Ø à 11ØØ On place les codes ASCII des caractères que vous avez tapés.

111Ø On ajoute à la fin de la ligne un octet nul, puis l'on calcule l'adresse de début de la prochaine ligne.

112Ø On incrémente le numéro de la ligne, et l'on va lire une autre ligne en 1Ø2Ø.

113Ø Cette ligne est exécutée lorsque l'on sort. DOKE A,Ø ajoute deux octets nuls à la suite de la dernière ligne, puis on met en #9C l'adresse de l'octet suivant.

2.6. COMMENT ACCÉLÉRER UN PROGRAMME BASIC

Dans tous les jeux d'action, il est souvent souhaitable d'accélérer au maximum les programmes. Voici donc quelques moyens d'accélérer l'exécution d'un programme Basic. Ces conseils sont valables pour la majorité des Basics, et donc en particulier pour celui de l'ORIC.

— Tout d'abord, et ceci est le plus important, remplacer toutes les constantes par des variables.

Exemples :

```
100 X=COS(0.0174532925*T)
```

peut être remplacé par :

```
10 CDR=O.0174532925
      ..........
100 X=COS(CDR*T)
```

Il est en effet beaucoup plus rapide de rechercher en mémoire la valeur d'une variable que de convertir une chaîne de chiffres en un réel.

— A chaque fois que l'on utilise une variable, celle-ci est cherchée en mémoire en partant de l'adresse stockée en #9C-#9D. Pour diminuer ce temps de recherche, on peut :

* diminuer le nombre total des variables, en réutilisant au maximum les mêmes noms. Par exemple, utiliser autant que possible la même variable pour toutes les boucles FOR ... NEXT.

* placer au début de la zone des variables celles qui sont utilisées le plus fréquemment, par exemple celles qui sont utilisées à l'intérieur des boucles. Pour cela, utiliser une fois ces variables au début du programme. Si vous ajoutez par exemple la ligne :

```
1 I=0:J=I:K=I:L=I
```

La variable I sera la première dans la zone des variables simples et sera donc retrouvée instantanément à chaque fois que l'on voudra l'utiliser. On trouvera ensuite la variable J, puis K, puis L.

— Se méfier des GOTO et GOSUB. En effet, cela prend du temps de retrouver un numéro de ligne dans un programme. Si un sous-programme est utilisé fréquemment, on a intérêt à le placer au *début du programme.*

```
  1 GOTO 1ØØØ REM VERS PROGRAMME
 1Ø REM S-PROGRAMME
          ..........

          ..........
1ØØØ REM DEBUT DU PROGRAMME
          ..........

          ..........
```

Dans la mesure du possible, remplacer les GOTO par des boucles FOR ... NEXT ou REPEAT ... UNTIL.

— Lorsque cela est possible, simplifiez ou supprimez les tests.

Exemples :

```
3Ø IF V<> Ø THEN 1ØØ
```

peut être remplacé par :

```
3Ø IF V THEN 1ØØ
1Ø IF A$=CHR$(8) THEN X=X-1
2Ø IF A$=CHR$(9) THEN X=X+1
```

peut être remplacé par la seule ligne :

```
1Ø X=X+(A$=CHR$(8))-(A$=CHR$(9))
```

En effet, (A$=CHR$(8)) est une expression logique qui vaut −1 si elle est vraie, Ø sinon. De même (A$=CHR$(9)) vaut −1 si elle est vraie et Ø sinon.

— Diminuez le nombre de lignes en mettant plusieurs instructions par ligne. Pour pouvoir mettre plus d'instructions par ligne, supprimez tous les blancs, ne conservez que les deux premiers caractères des noms de variables.

— Utilisez les variables entières. Évitez les tableaux.

— Supprimez les REM.

— Enfin, réécrivez en assembleur les parties critiques du jeu. C'est difficile, mais cela donne des résultats bien plus spectaculaires que tout ce qui précède.

2.7 UN PETIT EXEMPLE DE SOUS-PROGRAMME ASSEMBLEUR

Ce programme ne fait pas grand-chose, excepté montrer comment ajouter un sous-programme assembleur simple à un programme Basic. On trace un petit dessin au bas de l'écran HIRES, et on le déplace ensuite vers le haut de l'écran à l'aide du sous-programme assembleur.

On trouve un programme principal, de 1Ø à 14Ø, puis en 1ØØØ un sous-programme qui trace un petit dessin, puis en 2ØØØ un sous-programme qui sert à amener en mémoire les instructions 65Ø2. Nous allons d'abord regarder ces instructions machine.

Le sous-programme 65Ø2 déplace d'une ligne vers le haut une partie rectangulaire de l'écran HIRES. Comme chaque ligne est constituée de 4Ø octets, il suffit de déplacer de 4Ø les octets qui correspondent au rectangle.

En entrée du sous-programme, on a :

— en 1-2, l'adresse-mémoire de l'octet qui correspond au coin supérieur gauche du rectangle à déplacer.

— en 3, le nombre de lignes du rectangle.

— en 4, le nombre d'octets de chacune des lignes du rectangle.

En sortie du sous-programme, on a déplacé vers le haut d'une ligne le rectangle et :

— en 1-2, l'adresse-mémoire qui correspond au coin supérieur gauche du rectangle qui a été déplacé. C'est-à-dire l'adresse que l'on avait en entrée moins 4Ø.

— en 3, ce nombre n'a pas été modifié.

— en 4, ce nombre n'a pas été modifié.

— 5 et 6 sont modifiés.

Ce sous-programme DHAUT appelle lui-même deux sous-programmes NXLIG et PLIG qui ajoutent et retranchent respectivement 4Ø à l'adresse contenue en 1-2. Ce nombre 4Ø est celui des octets constituant une ligne écran, ils permettent donc de passer respectivement à la ligne suivante (NXLIG) et à la ligne précédente (PLIG).

```
A6  Ø3       DHAUT   LDX   3        Init. compteur de lignes.
2Ø  XX  XX           JSR   PLIG     Oter 4Ø à (1-2).
A5  Ø1               LDA   1        Sauvegarder 1
48                   PHA            dans la pile.
A5  Ø2               LDA   2        Sauvegarder 2
48                   PHA            dans la pile.
A4  Ø4       TLIGNE  LDY   4        Init. compteur d'octets par ligne.
A5  Ø1               LDA   1        Recopier 1
85  Ø5               STA   5        en 5, puis
A5  Ø2               LDA   2        recopier 2 en
85  Ø6               STA   5        6.
2Ø  XX  XX           JSR   NXLIG    Ajouter 4Ø à (1-2).
B1  Ø1       TRCARA  LDA   (1), Y   Prendre un octet et
91  Ø5               STA   (5), Y   le recopier, 4Ø octets avant.
88                   DEY            Décrémenter compteur d'octets.
1Ø  F9               BPL   TRCARA   Si compteur positif, continuer ligne
CA                   DEX            Sinon, décrémenter compteur lignes
1Ø  E9               BPL   TLIGNE   Si compteur positif, continuer.
68                   PLA            Terminé. Récupérer
85  Ø2               STA   2        2 et
68                   PLA            récupérer
85  Ø1               STA   1        1 puis
6Ø                   RTS            retour au programme Basic.


A5  Ø1       NXTLIG  LDA   1        Prendre octet de poid faible en 1
18                   CLC            préparer une addition
69  28               ADC   # 4Ø     ajouter 4Ø et
85  Ø1               STA   1        ranger le résultat.
9Ø  Ø2               BCC   RTS 1    Si pas de retenue, retour
E6  Ø2               INC   2        Sinon, propager la retenue
6Ø           RTS 1   RTS            Fin.


A5  Ø1       PLIG    LDA   1        Prendre octet de poids faible en 1
38                   SEC            préparer une soustraction
E9  28               SBC   # 4Ø     retrancher 4Ø et
85  Ø1               STA   1        ranger le résultat.
BØ  Ø2               BCS   RTS 2    Si il y a retenue, retour
C6  Ø2               DEC   2        Sinon, décrémenter poids fort
6Ø           RTS 2   RTS            Fin.
```

Ce sous-programme est mis en mémoire par les lignes 2ØØØ à 2Ø7Ø.

2Ø1Ø A est l'adresse-mémoire où l'on va placer les codes. #9CØØ correspond au début de la zone de stockage des caractères semi-graphiques en mode HIRES. Comme le programme n'utilise pas les

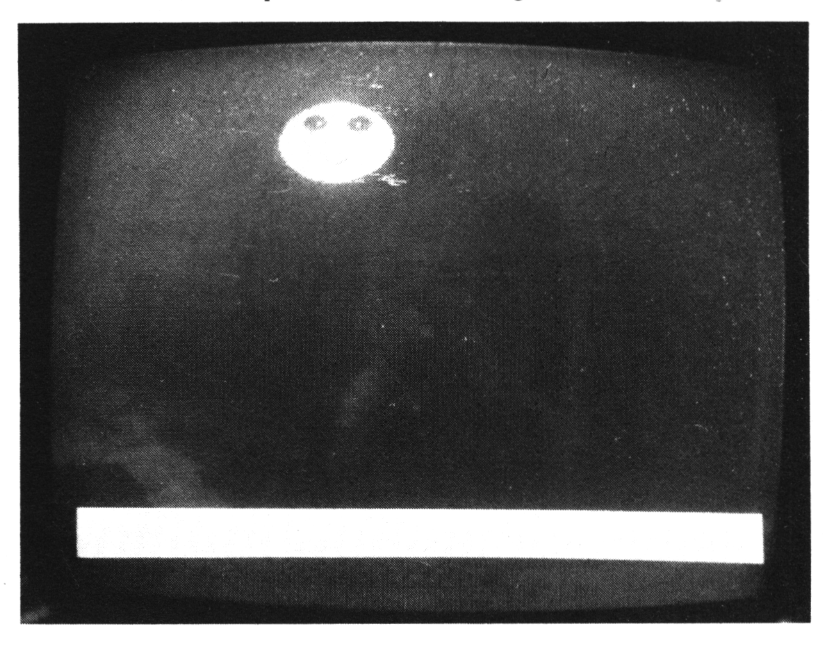
caractères semi-graphiques, on n'hésite pas à les écraser, car on a entre #9CØØ et #AØØØ, 1 Koctet de mémoire dans lequel le Basic ne va pas écrire, tant que l'on ne frappe pas reset.

2Ø2Ø à 2Ø4Ø Les 65 octets sont placés en mémoire à partir de l'adresse A.

2Ø5Ø Le sous-programme DHAUT est le premier, son adresse DH est égale à A. Comme DHAUT fait 41 octets, le second sous-programme NXTLIG débute à l'adresse NX=DH+41. Et comme NXTLIG fait 12 octets, le troisième, PLIG, débute donc à l'adresse PL=NX+12.

2Ø6Ø Après avoir calculé les adresses DH, NX et PL des sous-programmes, on peut ajouter au sous-programme les adresses de PLIG et NXTLIG, en DH+3 et DH+22 respectivement.

Pour utiliser DHAUT, on commence par placer en (1-2) l'adresse de l'octet qui correspond au coin supérieur gauche du rectangle à déplacer, ce qui est fait en 6Ø. En 7Ø, on place en 3 le nombre de lignes du rectangle, 41, et on place en 4 le nombre d'octets par ligne de ce rectangle, soit 9. Le sous-programme est appelé à la ligne 11Ø par CALL DH. Cet ordre est situé dans une boucle, le sous-programme est appelé 15Ø fois de suite, on déplace donc de 15Ø lignes vers le haut le dessin.



Pour finir, voici le listing du programme:

```
10 REM FACE DE LUNE******************
20 HIRES
30 GOSUB 2000
40 :
50 X=50+150*RND(1):Y=170
60 DOKE 1,#A000+(Y-20)*40+X/6-5
70 POKE 3,41:POKE 4,9
80 MUSIC 1,1,1,0:PLAY 1,0,3,100
90 GOSUB 1000
100 FOR I=1 TO 150
110 CALL DH:SOUND 1,200-I,15
120 NEXT
130 HIRES:GOTO 50
140 :
1000 REM DESSIN----------------------
1010 CURSET X,Y,3
1020 FOR R=1 TO 20:CIRCLE R,1:NEXT
1030 FOR I=-1 TO 1 STEP 2
1040 CURSET X+I*8,Y-10,0
1050 FOR J=1 TO 5:CIRCLE J,0:NEXT J
1060 CURSET X+I*7,Y-10,1
1070 FOR J=0 TO 6
1080 CURSET X+I*J,Y-J^2/6+10,0
1090 CURSET X+I*J,Y-J^2/6+11,0
1100 NEXT J
1110 NEXT I
1120 CURSET X-26,Y-20,3
1130 FILL 40,1,1+7*RND(1)
1140 RETURN
2000 REM SOUS-PROGRAMMES 6502--------
2010 A=#9C00
2020 FOR I=A TO A+64
2030 READ V:POKE I,V
2040 NEXT I
2050 DH=A:NX=DH+41:PL=NX+12
2060 DOKE DH+3,PL:DOKE DH+22,NX
2070 RETURN
2080 :
2090 REM DHAUT.......................
```

```
2100 DATA#A6,#03,#20,#FF,#FF,#A5,#01
2110 DATA#48,#A5,#02,#48,#A4,#04,#A5
2120 DATA#01,#85,#05,#A5,#02,#85,#06
2130 DATA#20,#FF,#FF,#B1,#01,#91,#05
2140 DATA#88,#10,#F9,#CA,#10,#E9,#68
2150 DATA#85,#02,#68,#85,#01,#60
2160 :
2170 REM NXTLIG......................
2180 DATA#A5,#01,#18,#69,#28,#85,#01
2190 DATA#90,#02,#E6,#02,#60
2200 :
2210 REM PLIG........................
2220 DATA#A5,#01,#38,#E9,#28,#85,#01
2230 DATA#B0,#02,#C6,#02,#60
```

Annexes

3.1 TABLE ASCII DE L'ORIC/ATMOS

Code ASCII	*Touche*	*Effet de la touche ou de PRINT CHR$(code)*	*Effet de PLOT X,Y,code*
Ø	pas de touche	Aucun	Tous les caractères affichés sur la ligne Y et situés après la colonne X sont noirs.
1	CTRL-A	CTRL-A relit un caractère sur l'écran.	Tous les caractères affichés sur la ligne Y et situés après la colonne X sont rouges.
2	CTRL-B	Aucun	Tous les caractères affichés sur la ligne Y et situés après la colonne X sont verts.
3	CTRL-C	CTRL-C interrompt un programme basic.	Tous les caractères affichés sur la ligne Y et situés après la colonne X sont jaunes.
4	CTRL-D	Provoque un affichage double. Les caractères suivants seront affichés deux fois chacun. Pour annuler, utiliser de nouveau CHR$(4) ou CTRL-D	Tous les caractères affichés sur la ligne Y et situés après la colonne X sont bleus.
5	CTRL-E	Aucun	Tous les caractères affichés sur la ligne Y et situés après la colonne X sont violets.

Code ASCII	*Touche*	*Effet de la touche ou de PRINT CHR$(code)*	*Effet de PLOT X,Y,code*
6	CTRL-F	Supprime le "plop" que l'on entend lorsque l'on frappe une touche. Pour retrouver le bruit, réutiliser CHR$(6) ou CTRL-F.	Tous les caractères affichés sur la ligne Y et situés après la colonne X sont bleus ciels (cyan).
7	CTRL-G	Fait retentir PING.	Tous les caractères affichés sur la ligne Y et situés après la colonne X sont blancs.
8	CTRL-H ou ←	Déplace le curseur un cran à gauche.	Tous les caractères affichés sur la ligne Y et situés après la colonne X seront : — en hauteur simple, — en affichage fixe, — jeu de caractères standard.
9	CTRL-I ou →	Déplace le curseur un cran à droite.	Tous les caractères affichés sur la ligne Y et situés après la colonne Y seront : — en hauteur simple, — en affichage fixe, — jeu de caractères semi-graphique.
1Ø	CTRL-J ou ↓	Déplace le curseur un cran vers le bas.	Tous les caractères affichés sur la ligne Y et situés après la colonne X seront : — en double hauteur, — en affichage fixe, — jeu de caractères standard.
11	CTRL-K ou ↑	Déplace le curseur un cran vers le haut.	Tous les caractères affichés sur la ligne Y et situés après la colonne X seront : — en double hauteur, — en affichage fixe, — jeu de caractères semi-graphique.
12	CTRL-L	Efface l'écran.	Tous les caractères affichés sur la ligne Y et situés après la colonne X seront : — en simple hauteur, — en affichage clignotant, — jeu de caractères standard.

Code ASCII	Touche	Effet de la touche ou de PRINT CHR$(code)	Effet de PLOT X,Y,code
13	CTRL-M ou RETURN	Envoie le curseur au début de la ligne suivante.	Tous les caractères affichés sur la ligne Y et situés après la colonne X seront : — en simple hauteur, — en affichage clignotant, — jeu de caractères semi-graphique.
14	CTRL-N	Efface la ligne ou se trouve le curseur.	Tous les caractères affichés sur la ligne Y et situés après la colonne X seront : — en double hauteur, — en affichage clignotant, — jeu de caractères standard.
15	CTRL-O	Aucun.	Tous les caractères affichés sur la ligne Y et situés après la colonne X seront : — en double hauteur, — en affichage clignotant, — jeu de caractère semi-graphique.
16	CTRL-P	Aucun.	Le fond (PAPER) de la ligne Y, à partir de la colonne X, sera noir.
17	CTRL-Q	Supprime le curseur. Pour le retrouver, réutiliser CHR$(17) ou CTRL-Q.	Le fond (PAPER) de la ligne Y, à partir de la colonne X, sera rouge.
18	CTRL-R	Aucun.	Le fond (PAPER) de la ligne Y, à partir de la colonne X, sera vert.
19	CTRL-S	Supprime l'affichage. Pour le rétablir, réutiliser CHR$(17) ou CTRL-S.	Le fond (PAPER) de la ligne Y, à partir de la colonne X, sera jaune.
2Ø	CTRL-T	Passage en minuscules. Pour obtenir une majuscule, utilisez SHIFT. Pour revenir en majuscules, réutilisez CHR$(20) ou CTRL-T.	Le fond (PAPER) de la ligne Y, à partir de la colonne X, sera bleu.
21	CTRL-U	Aucun.	Le fond (PAPER) de la ligne Y, à partir de la colonne X, sera violet.

Code ASCII	Touche	Effet de la touche ou de PRINT CHR$(code)	Effet de PLOT X,Y,code
22	CTRL-V	Aucun.	Le fond (PAPER) de la ligne Y, à partir de la colonne X, sera cyan.
23	CTRL-W	Aucun.	Le fond (PAPER) de la ligne Y, à partir de la colonne Y, sera blanc.
24	CTRL-X	CTRL-X annule la ligne en train d'être frappée.	Passage en texte 60 Hz. Ne pas utiliser, perturbe l'affichage.
25	CTRL-Y	Aucun.	Passage en texte 60 Hz. Ne pas utiliser.
26	CTRL-Z	A la même action que ESC. Voir ci-dessous.	Passage en texte 50 Hz, rend l'affichage normal.
27	CTRL-[ou ESC	Ce caractère n'est pas affiché, et ne déplace pas le curseur. Par contre, le caractère suivant sera affiché après que ses bits b_5 et b_6 aient été mis à zéro.	Passage en texte 50 Hz, rend l'affichage normal.
28	CTRL-/	Aucun.	Passage en graphique 60 Hz. Ne pas utiliser, perturbe l'affichage.
29	CTRL-]	Normalement, lorsque l'on frappe sur une touche ou que l'on exécute PRINT, aucun caractère n'est affiché dans les deux premières colonnes de l'écran. Le caractère permet d'utiliser ces colonnes. Mais alors les textes sont blancs sur fond noir, car les caractères commandant la couleur de l'écran et du texte, qui se trouvent normalement dans ces deux colonnes, sont écrasés. Réutilisez CHR$(29) ou CTRL-] pour revenir comme avant.	Passage en graphique 60 Hz. Ne pas utiliser, perturber l'affichage.

Code ASCII	*Touche*	*Effet de la touche ou de PRINT CHR$(code)*	*Effet de PLOT X,Y,code*
3Ø	Pas de touche	CHR$(3Ø) renvoie le curseur dans le coin supérieur gauche de l'écran, qui n'est pas effacé.	Passage en graphique 50 Hz. Affichage normal.
31	CTRL-DEL	Aucun.	Passage en graphique 50 Hz. Affichage normal.
32	barre d'espacement	Affiche un espace, un blanc	
33	!	Affiche un !	
34	"	Affiche un " (guillemet)	
35	#	Affiche un # (dièse)	
36	$	Affiche un $ (dollar)	
37	%	Affiche un %	
38	&	Affiche un &	
39	'	Affiche un ' (apostrophe)	
4Ø	(	Affiche un (	
41	)	Affiche un)	
42	*	Affiche un * (signe de multiplication)	
43	+	Affiche un +	
44	,	Affiche un ,	
45	—	Affiche un —	
46	.	Affiche un .	
47	/	Affiche un / (signe de division, slash)	
48	Ø	Affiche un Ø	
49	1	Affiche un 1	
5Ø	2	Affiche un 2	
51	3	Affiche un 3	
52	4	Affiche un 4	
53	5	Affiche un 5	
54	6	Affiche un 6	
55	7	Affiche un 7	
56	8	Affiche un 8	

Code ASCII	*Touche*	*Effet*	
57	9	Affiche un 9	
58	:	Affiche un :	(colon)
59	;	Affiche un ;	(semi-colon)
6Ø	<	Affiche un <	
61	=	Affiche un =	
62	>	Affiche un >	
63	?	Affiche un ?	
64	@	Affiche un @	
65	A	Affiche un A	
66	B	Affiche un B	
67	C	Affiche un C	
68	D	Affiche un D	
69	E	Affiche un E	
7Ø	F	Affiche un F	
71	G	Affiche un G	
72	H	Affiche un H	
73	I	Affiche un I	
74	J	Affiche un J	
75	K	Affiche un K	
76	L	Affiche un L	
77	M	Affiche un M	
78	N	Affiche un N	
79	O	Affiche un O	
8Ø	P	Affiche un P	
81	Q	Affiche un Q	
82	R	Affiche un R	
83	S	Affiche un S	
84	T	Affiche un T	
85	U	Affiche un U	
86	V	Affiche un V	
87	W	Affiche un W	

Code ASCII	*Touche*	*Effet*	
88	X	Affiche un X	
89	Y	Affiche un Y	
90	Z	Affiche un Z	
91	[	Affiche un [	(crochet ouvrant)
92	\	Affiche un \	(backslash)
93	]	Affiche un]	(crochet fermant)
94	^	Affiche un	↑
95	£	Affiche un £	(livre)
96	pas de touche	Affiche un ©	
97	A	Affiche un a	Pour obtenir les
98	B	Affiche un b	minuscules au clavier,
99	C	Affiche un c	frapper d'abord
100	D	Affiche un d	CTRL-T
101	E	Affiche un e	
102	F	Affiche un f	
103	G	Affiche un g	
104	H	Affiche un h	
105	I	Affiche un i	
106	J	Affiche un j	
107	K	Affiche un k	
108	L	Affiche un l	
109	M	Affiche un m	
110	N	Affiche un n	
111	O	Affiche un o	
112	P	Affiche un p	
113	Q	Affiche un q	
114	R	Affiche un r	
115	S	Affiche un s	
116	T	Affiche un t	
117	U	Affiche un u	

Code ASCII	*Touche*	*Effet*
118	V	Affiche un v
119	W	Affiche un w
12Ø	X	Affiche un x
121	Y	Affiche un y
122	Z	Affiche un z
123	{	Affiche un
124	I	Affiche un I
125	}	Affiche un
126	pas de touche	Affiche un carré clair
127	DEL	DEL annule un caractère CHR$(127) affiche un carré sombre.
		Les codes 128 à 255 provoquent les mêmes effets que les codes de Ø à 127. Avec deux petites différences : — les codes 128 à 159 sont affichés, alors que les codes Ø à 31 ne le sont pas. CHR$(128) a le effet que ESC , CHR$(129) a le même effet que ESCA, ..., etc., — les caractères correspondant aux codes de 128 à 255 sont affichés en vidéo inverse, lorsqu'on les utilise avec PLOT.

3.2 TABLE DES ATTRIBUTS VIDÉO

Les codes ASCII Ø à 31 ont une action spéciale lorsqu'on les place dans la zone mémoire correspondant à l'écran TEXT ou HIRES.

Ils permettent de choisir la couleur de l'encre, du fond, le jeu de caractère, normal ou semi-graphique, la taille des caractères et le clignotement.

Pour placer un code dans la mémoire-écran, vous pouvez :

— Exécuter POKE adresse, code

— Exécuter PLOT X,Y,CHR$(code) ou PLOT X,Y,code

— Exécuter PRINT CHR$(27); CHR$(code+64)

— Frapper la touche ESC puis la touche dont le code ASCII vaut code +64.

Par exemple, pour code=Ø, code+64 vaut 64 et CHR$(code+64) vaut "@". Pour code=1, code+64 vaut 65 et CHR$(code+64) vaut "A".

Chacun de ces codes est actif depuis l'endroit où il se trouve jusqu'à la fin de sa ligne.

Attributs de Ø à 15

Avec POKE ou PLOT code	*Avec PRINT*	*Depuis le clavier*	*Action sur le système vidéo*
Ø	PRINT CHR$(27);"@"	ESC @	encre noire
1	PRINT CHR$(27);"A"	ESC A	encre rouge
2	PRINT CHR$(27);"B"	ESC B	encre verte
3	PRINT CHR$(27);"C"	ESC C	encre jaune
4	PRINT CHR$(27);"D"	ESC D	encre bleue
5	PRINT CHR$(27);"E"	ESC E	encre magenta
6	PRINT CHR$(27);"F"	ESC F	encre cyan
7	PRINT CHR$(27);"G"	ESC G	encre blanche
8	PRINT CHR$(27);"H"	ESC H	simple hauteur, fixe, normal
9	PRINT CHR$(27);"I"	ESC I	simple hauteur, fixe, semi-graphique

1Ø	PRINT CHR$(27);"J"	ESC J	double hauteur, fixe, normal
11	PRINT CHR$(27);"K"	ESC K	double hauteur, fixe, semi-graphique
12	PRINT CHR$(27);"L"	ESC L	simple hauteur, clignotant, normal
13	PRINT CHR$(27);"M"	ESC M	simple hauteur, clignotant, semi-graphique
14	PRINT CHR$(27);"N"	ESC N	double hauteur, clignotant, normal
15	PRINT CHR$(27);"O"	ESC O	double hauteur, clignotant, semi-graphique

Les codes de Ø à 7 permettent de fixer la couleur de l'encre, tandis que les codes 8 à 15 sélectionnent le type de caractère employé, un caractère peut en effet être :

— normal ou semi-graphique

— simple hauteur ou double hauteur

— fixe ou clignotant

Attributs de 16 à 31

Avec POKE ou PLOT code	*Avec PRINT*	*Depuis le clavier*	*Action sur le système vidéo*
16	PRINT CHR$(27);"P"	ESC P	écran noir
17	PRINT CHR$(27);"Q"	ESC Q	écran rouge
18	PRINT CHR$(27);"R"	ESC R	écran vert

19	PRINT CHR$(27);"S"	ESC S	écran jaune
2Ø	PRINT CHR$(27);"T"	ESC T	écran bleu
21	PRINT CHR$(27);"U"	ESC U	écran magenta
22	PRINT CHR$(27);"V"	ESC V	écran cyan
23	PRINT CHR$(27);"W"	ESC W	écran blanc
24	PRINT CHR$(27);"X"	ESC X	mode texte en 60 Hz
25	PRINT CHR$(27);"Y"	ESC Y	mode texte en 60 Hz
26	PRINT CHR$(27);"Z"	ESC Z	mode texte en 50 Hz
27	PRINT CHR$(27);" { "	ESC {	mode texte en 50 Hz
28	PRINT CHR$(27);" | "	ESC |	mode graphique en 60 Hz
29	PRINT CHR$(27);" } "	ESC }	mode graphique en 60 Hz
3Ø	PRINT CHR$(27); CHR$(3Ø)	pas de touche	mode graphique en 50 Hz
31	PRINT CHR$(27); CHR$(31)	ESC DEL	mode graphique en 50 Hz

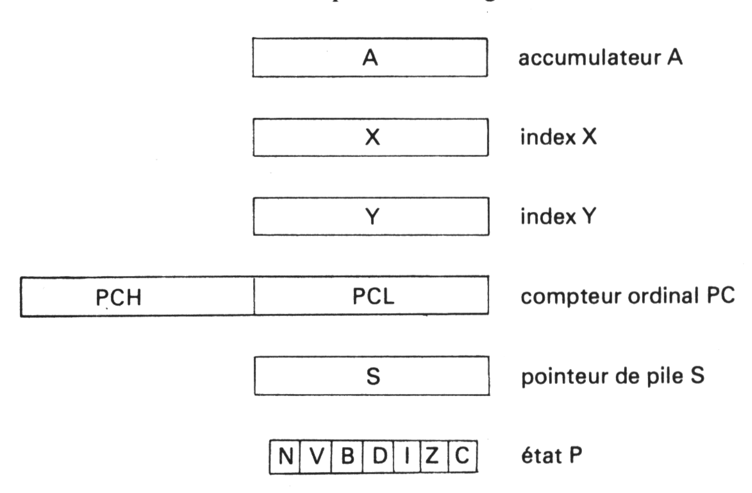
Les codes de 16 à 23 permettent de fixer la couleur de l'encre, tandis que les codes 24 à 31 commandent le mode vidéo. On peut :

— être en mode texte (TEXT et LORES) ou en mode graphique (HIRES)

— balayer l'écran 50 fois ou 60 fois par seconde. En France, le balayage se fait en 50 Hz.

3.3 LE MICROPROCESSEUR 65Ø2

Le microprocesseur 65Ø2, qui équipe votre ORIC, est capable de manipuler, de traiter des données de 8 bits (octets), et engendre des adresses de 16 bits. Il dispose de 7 registres :



* A : Ce registre de 8 bits contient le résultat de toutes les opérations, logiques ou arithmétiques.

* X et Y : Ces deux registres de 8 bits peuvent servir de compteur ou servent à construire une adresse.

* PC : Ce registre de 16 bits contient l'adresse de la prochaine instruction à exécuter.

* S : Ce registre de 8 bits contient l'octet de poids faible du pointeur de pile. L'octet de poids fort vaut 1, car par construction du 65Ø2, la pile est située entre 1ØØ et 1FF (adresses hexadécimales).

* P : Ce registre de 7 bits contient 7 indicateurs.

  N Indicateur de résultat négatif.

  V Indicateur de débordement (Overflow).

  B Indicateur de BREAK.

D	Indicateur de mode de calcul décimal.
I	Indicateur d'interruptions masquées.
Z	Indicateur de résultat nul (Zéro).
C	Retenue (Carry).

Le 65Ø2 dispose de 13 modes d'adressage qui sont :

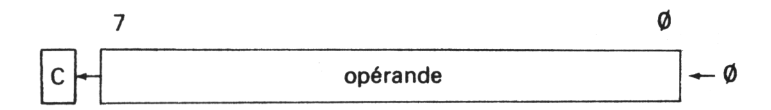
— Immédiat	ex : LDA #FF
— Page zéro	LDA 1
— Page zéro indexé par X	LDA 1,X
— Absolu	LDA 1ØØØ
— Absolu indexé par X	LDA 1ØØØ,X
— Absolu indexé par Y	LDA 1ØØØ,Y
— Indirect pré-indexé par X	LDA (1ØØØ,X)
— Indirect post-indexé par Y	LDA (1ØØØ),Y
— Accumulateur	ROL
— Implicite	TAX
— Relatif	BNE 1ØØØ
— Indirect	JMP (1ØØØ)
— Page zéro indexé par Y	LDX 1,Y

Le 65Ø2 dispose de 56 instructions qui sont :

ADC L'opérande est ajouté au bit C de retenue et à l'accumulateur, le résultat est dans l'accumulateur.

AND On effectue le ET logique bit à bit de l'accumulateur et de l'opérande, le résultat est dans l'accumulateur.

ASL L'opérande est décalé d'un bit vers la gauche.



BCC Branchement relatif lorsque C = Ø (pas de retenue).

BCS Branchement relatif lorsque C = 1 (retenue).

BEQ Branchement relatif lorsque Z = 1 (résultat nul).

BIT Positionne les indicateurs suivant le résultat de la comparaison bit à bit de l'opérande et de l'accumulateur.

BMI Branchement relatif lorsque N = 1 (résultat négatif).

BNE Branchement relatif lorsque Z = Ø (résultat non nul).

BPL Branchement relatif lorsque N = Ø (résultat positif).

BRK Break, déroutement software.

BVC Branchement relatif lorsque V = Ø (pas d'overflow).

BVS Branchement relatif lorsque V = 1 (overflow).

CLC Positionne à Ø l'indicateur C (carry).

CLD Positionne à Ø l'indicateur D (mode décimal).

CLI Positionne à Ø l'indicateur I (interruptions démasquées).

CLV Positionne à Ø l'indicateur V (overflow).

CMP Compare l'accumulateur et l'opérande et positionne en conséquence les indicateurs.

CPX Compare le registre X et l'opérande, positionne les indicateurs.

CPY Compare le registre Y et l'opérande, positionne les indicateurs.

DEC Décrémente d'une unité l'opérande.

DEX Décrémente d'une unité le registre X.

DEY Décrémente d'une unité le registre Y.

EOR "OU" exclusif bit à bit de l'opérande et l'accumulateur, le résultat est dans l'accumulateur.

INC Incrémente l'opérande d'une unité.

INX Incrémente le registre X d'une unité.

INY Incrémente le registre Y d'une unité.

JMP Branchement absolu inconditionnel.

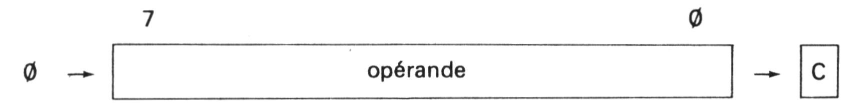
JSR Appel d'un sous-programme.

LDA Charge l'opérande dans l'accumulateur.

LDX Charge l'opérande dans le registre X.

LDY Charge l'opérande dans le registre Y.

LSR L'opérande est décalé d'un bit vers la droite.



NOP Pas d'opération. Ne fait rien.

ORA "OU" bit à bit de l'opérande et l'accumulateur, le résultat est dans l'accumulateur.

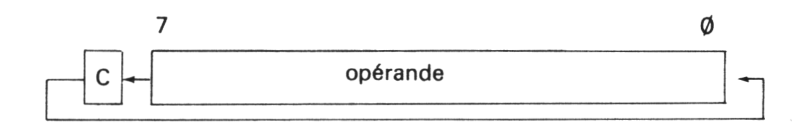
PHA Empile l'accumulateur.

PHP Empile les indicateurs (registre P).

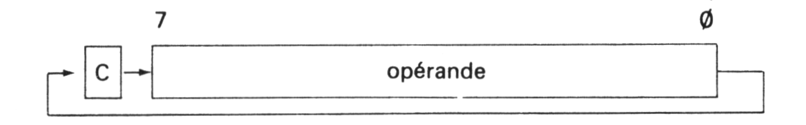
PLA Dépile l'accumulateur.

PLP Dépile les indicateurs (registre P).

ROL Rotation d'un bit vers la gauche au travers de C de l'opérande.



ROR Rotation d'un bit vers la droite au travers de C de l'opérande.



RTI Retour de sous-programme de traitement d'interrruption.

RTS Retour de sous-programme.

SBC Retranche à l'accumulateur l'opérande et la carry.

SEC Positionne à 1 l'indicateur C (carry).

SED Positionne à 1 l'indicateur D (mode décimal).

SEI Positionne à 1 l'indicateur I (masque les interruptions).

STA	Range l'accumulateur en mémoire.
STX	Range le registre X en mémoire.
STY	Range le registre Y en mémoire.
TAX	Recopie A dans X.
TAY	Recopie A dans Y.
TSX	Recopie S (pointeur de pile) dans X.
TXA	Recopie X dans A.
TXS	Recopie X dans S.
TYA	Recopie Y dans A.

Mnémonique	Opération	Modes d'adressages	Syntaxe assembleur	Code HEXA	Nombre d'octets	N	Z	C	I	D	V
ADC	A+M+C→A	immédiat	ADC #oper	69	2	N	Z	C	●	●	V
		page zéro	ADC oper	65	2						
		page zéro,X	ADC oper,X	75	2						
		absolu	ADC oper	6D	3						
		absolu,X	ADC oper,X	7D	3						
		absolu,Y	ADC oper,Y	79	3						
		(indirect,X)	ADC (oper,X)	61	2						
		(indirect),Y	ADC (oper),Y	71	2						
AND	A.M→A	immédiat	AND #oper	29	2	N	Z	●	●	●	●
		page zéro	AND oper	25	2						
	"ET" bit à bit	page zéro,X	AND oper,X	35	2						
		absolu	AND oper	2D	3						
		absolu,X	AND oper,X	3D	3						
		absolu,Y	AND oper,Y	39	3						
		(indirect,X)	AND (oper,X)	21	2						
		(indirect),Y	AND (oper),Y	31	2						
ASL	décalage à gauche	accumulateur	ASL	ØA	1	N	Z	C	●	●	●
		page zéro	ASL oper	Ø6	2						
		page zéro,X	ASL oper,X	16	2						
		absolu	ASL oper	ØE	3						
		absolu,X	ASL oper,X	1E	3						
BCC	branche si C = Ø	relatif	BCC oper	9Ø	2	●	●	●	●	●	●

Mnémonique	*Opération*	*Modes d'adressages*	*Syntaxe assembleur*	*Code* HEXA	*Nombre d'octets*	*Indicateurs modifiés* N	Z	C	I	D	V
BCS	branche si C = 1	relatif	BCS oper	BØ	2	●	●	●	●	●	●
BEQ	branche si Z = 1	relatif	BEQ oper	FØ	2	●	●	●	●	●	●
BIT	A.M $M_7 \rightarrow N$ $M_6 \rightarrow V$	page zéro absolu	BIT oper BIT oper	24 2C	2 3	M_7	Z	●	●	●	M_6
BMI	branche si N = 1	relatif	BMI oper	3Ø	2	●	●	●	●	●	●
BNE	branche si Z = Ø	relatif	BNE oper	DØ	2	●	●	●	●	●	●
BPL	branche si N = Ø	relatif	BPL oper	1Ø	2	●	●	●	●	●	●
BRK	déroutement	implicite	BRK	ØØ	1	●	●	●	1	●	●
BVC	branche si V = Ø	relatif	BVC oper	5Ø	2	●	●	●	●	●	●
BVS	branche si V = 1	relatif	BVS oper	7Ø	2	●	●	●	●	●	●
CLC	Ø→C	implicite	CLC	18	1	●	●	Ø	●	●	●

Mnémonique	Opération	Modes d'adressages	Syntaxe assembleur	Code HEXA	Nombre d'octets	Indicateurs modifiés N Z C I D V
CLD	Ø→D	implicite	CLD	D8	1	● ● ● ● Ø ●
CLI	Ø→I	implicite	CLI	58	1	● ● ● Ø ● ●
CLV	Ø→V	implicite	CLV	B8	1	● ● ● ● ● Ø
CMP	positionne les indicateurs suivant A—M	immédiat	CMP #oper	C9	2	N Z C ● ● ●
		page zéro	CMP oper	C5	2	
		page zéro,X	CMP oper,X	D5	2	
		absolu	CMP oper	CD	3	
		absolu,X	CMP oper,X	DD	3	
		absolu,Y	CMP oper,Y	D9	3	
		(indirect,X)	CMP (oper,X)	C1	2	
		(indirect),Y	CMP (oper),Y	D1	2	
CPX	positionne les indicateurs suivant X—M	immédiat	CPX #oper	EØ	2	N Z C ● ● ●
		page zéro	CPX oper	E4	2	
		absolu	CPX oper	EC	3	
CPY	positionne les indicateurs suivant Y—M	immédiat	CPY #oper	CØ	2	N Z C ● ● ●
		page zéro	CPY oper	C4	2	
		absolu	CPY oper	CC	3	

Mnémonique	*Opération*	*Modes d'adressages*	*Syntaxe assembleur*	*Code* HEXA	*Nombre d'octets*	*Indicateurs modifiés* N Z C I D V
DEC	M−1→M	page zéro page zéro,X absolu absolu,X	DEC oper DEC oper,X DEC oper DEC oper,X	C6 D6 CE DE	2 2 3 3	N Z ● ● ● ●
DEX	X−1→X	implicite	DEX	CA	1	N Z ● ● ● ●
DEY	Y−1→Y	implicite	DEY	88	1	N Z ● ● ● ●
EOR	A ⊕ M→A "OU" exclusif bit à bit	immédiat page zéro page zéro,X absolu absolu,X absolu,Y (indirect,X) (indirect),Y	EOR #oper EOR oper EOR oper,X EOR oper EOR oper,X EOR oper,Y EOR (oper,X) EOR (oper),Y	49 45 55 4D 5D 59 41 51	2 2 2 3 3 3 2 2	N Z ● ● ● ●
INC	M+1→M	page zéro page zéro,X absolu absolu,X	INC oper INC oper,X INC oper INC oper,X	E6 F6 EE FE	2 2 3 3	N Z ● ● ● ●
INX	X+1→X	implicite	INX	E8	1	N Z ● ● ● ●

Mnémonique	*Opération*	*Modes d'adressages*	*Syntaxe assembleur*	Code HEXA	*Nombre d'octets*	*Indicateurs modifiés* N	Z	C	I	D	V
INY	Y+1→Y	implicite	INY	C8	1	N	Z	●	●	●	●
JMP	(PC+1)→PCL (PC+2)→PCH	absolu	JMP oper	4C	3	●	●	●	●	●	●
		indirect	JMP (oper)	6C	3						
JSR	empile PC+2 (PC+1)→PCL (PC+2)→PCH	absolu	JSR oper	2Ø	3	●	●	●	●	●	●
LDA	M→A	immédiat	LDA #oper	A9	2	N	Z	●	●	●	●
		page zéro	LDA oper	A5	2						
		page zéro,X	LDA oper,X	B5	2						
		absolu	LDA oper	AD	3						
		absolu,X	LDA oper,X	BD	3						
		absolu,Y	LDA oper,Y	B9	3						
		(indirect,X)	LDA (oper,X)	A1	2						
		(indirect),Y	LDA (oper),Y	B1	2						
LDX	M→X	immédiat	LDX #oper	A2	2	N	Z	●	●	●	●
		page zéro	LDX oper	A6	2						
		page zéro,Y	LDX oper,Y	B6	2						
		absolu	LDX oper	AE	3						
		absolu,Y	LDX oper,Y	BE	3						

Mnémonique	*Opération*	*Modes d'adressages*	*Syntaxe assembleur*	Code HEXA	*Nombre d'octets*	*Indicateurs modifiés* N Z C I D V
LDY	M→Y	immédiat	LDY #oper	AØ	2	N Z ● ● ● ●
		page zéro	LDY oper	A4	2	
		page zéro,X	LDY oper,X	B4	2	
		absolu	LDY oper	AC	3	
		absolu,X	LDY oper,X	BC	3	
LSR	décalage d'un bit à droite	accumulateur	LSR	4A	1	Ø Z C ● ● ●
		page zéro	LSR oper	46	2	
		page zéro,X	LSR oper,X	56	2	
		absolu	LSR oper	4E	3	
		absolu,X	LSR oper,X	5E	3	
NOP	rien	implicite	NOP	EA	1	● ● ● ● ● ●
ORA	AVM→A "OU" bit à bit	immédiat	ORA #oper	Ø9	2	N Z ● ● ● ●
		page zéro	ORA oper	Ø5	2	
		page zéro,X	ORA oper,X	15	2	
		absolu	ORA oper	ØD	3	
		absolu,X	ORA oper,X	1D	3	
		absolu,Y	ORA oper,Y	19	3	
		(indirect,X)	ORA (oper,X)	Ø1	2	
		(indirect),Y	ORA (oper),Y	11	2	
PHA	empile A	implicite	PHA	48	1	● ● ● ● ● ●

Mnémonique	*Opération*	*Modes d'adressages*	*Syntaxe assembleur*	Code HEXA	*Nombre d'octets*	*Indicateurs modifiés* N Z C I D V
PHP	empile P	implicite	PHP	Ø8	1	● ● ● ● ● ●
PLA	dépile A	implicite	PLA	68	1	N Z ● ● ● ●
PLP	dépile P	implicite	PLP	28	1	N Z C I D V
ROL	Rotation d'un bit à gauche	accumulateur page zéro page zéro,X absolu absolu,X	ROL ROL oper ROL oper,X ROL oper ROL oper,X	2A 26 36 2E 3E	1 2 2 3 3	N Z C ● ● ●
ROR	Rotation d'un bit à droite	accumulateur page zéro page zéro,X absolu absolu,X	ROR ROR oper ROR oper,X ROR oper ROR oper,X	6A 66 76 6E 7E	1 2 2 3 3	N Z C ● ● ●
RTI	retour d'interruption	implicite	RTI	4Ø	1	N Z C I D V
RTS	retour de sous-programme	implicite	RTS	6Ø	1	● ● ● ● ● ●

Mnémonique	Opération	Modes d'adressages	Syntaxes assembleur	Code HEXA	Nombre d'octets	Indicateurs modifiés N	Z	C	I	D	V
SBC	A—M—C→A	immédiat	SBC #oper	E9	2	N	Z	C	●	●	V
		page zéro	SBC oper	E5	2						
		page zéro,X	SBC oper,X	F5	2						
		absolu	SBC oper	ED	3						
		absolu,X	SBC oper,X	FD	3						
		absolu,Y	SBC oper,Y	F9	3						
		(indirect,X)	SBC (oper,X)	E1	2						
		(indirect),Y	SBC (oper),Y	F1	2						
SEC	1→C	implicite	SEC	38	1	●	●	1	●	●	●
SED	1→D	implicite	SED	F8	1	●	●	●	●	1	●
SEI	1→I	implicite	SEI	78	1	●	●	●	1	●	●
STA	A→M	page zéro	STA oper	85	2	●	●	●	●	●	●
		page zéro,X	STA oper,X	95	2						
		absolu	STA oper	8D	3						
		absolu,X	STA oper,X	9D	3						
		absolu,Y	STA oper,Y	99	3						
		(indirect,X)	STA (oper,X)	81	2						
		(indirect),Y	STA (oper),Y	91	2						

Mnémonique	*Opération*	*Modes d'adressages*	*Syntaxes assembleur*	*Code* HEXA	*Nombre d'octets*	*Indicateurs modifiés* N Z C I D V
STX	X→M	page zéro page zéro,Y absolu	STX oper STX oper,Y STX oper	86 96 8E	2 2 3	● ● ● ● ● ●
STY	Y→M	page zéro page zéro,X absolu	STY oper STY oper,X STY oper	84 94 8C	2 2 3	● ● ● ● ● ●
TAX	A→X	implicite	TAX	AA	1	N Z ● ● ● ●
TAY	A→Y	implicite	TAY	A8	1	N Z ● ● ● ●
TSX	S→X	implicite	TSX	BA	1	N Z ● ● ● ●
TXA	X→A	implicite	TXA	8A	1	N Z ● ● ● ●
TXS	X→S	implicite	TXS	9A	1	● ● ● ● ● ●
TYA	Y→A	implicite	TYA	98	1	N Z ● ● ● ●

Imprimerie de la Manutention à Mayenne
Dépôt légal : juillet 1984
N° d'Éditeur : 4150

Après « ORIC-1 à la conquête des jeux », voici maintenant « ATMOS à la conquête des jeux », qui reprend l'ouvrage précédent en y ajoutant toutes les améliorations de l'ATMOS.

« ATMOS à la conquête des jeux » se propose de vous entraîner dans son univers de programmes, celui des jeux, des dessins en couleur, des sonorités synthétiques informatiques. Vous y rencontrerez une gentille petite chenille, six caméléons et quelques gloutons. Vous voyagerez dans l'espace intersidéral. Vous verrez la tour de Hanoï, un grand mur de briques. Vous vous mesurerez au Master-Mind.

Nul doute qu'après tout cela, vous désiriez mieux connaître votre ATMOS et, pourquoi pas, programmer vos jeux. Pour vous aider et vous guider dans ce labyrinthe aux 65536 adresses, « ATMOS à la conquête des Jeux » vous révèle le contenu de la mémoire de ATMOS, comment redéfinir les caractères, tous les trucs des modes HIRES et TEXT, et même, mais oui, comment fabriquer des lignes de programme Basic avec un autre programme...

Bon amusement !



EYROLLES

8706


EYROLLES
ATMOS A LA CONQUÊTE DES JEUX
J.Y. ASTIER